Dependency Graph Based Sentence Fusion and Compression

Vom Fachbereich Gesellschafts- und Geschichtswissenschaften
der Technischen Universitaet Darmstadt
zur Erlangung des Grades eines Doktors der Philosophie (Dr. phil.)
genehmigte Dissertation
von

M.A. Ekaterina (Katja) Filippova
aus Sankt-Petersburg (Russland)

Referent: Prof. Dr. Elke Teich
Koreferent: Mirella Lapata (PhD, Reader)

Einreichung: 25. Juni 2009
Prüfung: 9. Oktober 2009

D17

Darmstadt
2010

Abstract

The popularity of text summarization (TS) in the NLP community has been steadily increasing in recent years. This is not surprising given its practical utility: e.g., multi-document summarization systems would be of great use given the enormous amount of news published daily online. Although TS methods vary considerably, most of them share one important property: they are extractive, and the most common extraction unit is the sentence – that is, most TS systems build summaries from extracted sentences. The extractive strategy has a well-recognized drawback which is related to the fact that sentences pulled from different documents may overlap but also complement each other. As a consequence, extractive systems are often unable to produce summaries which are complete and non-redundant at the same time. Sentence fusion (Barzilay & McKeown, 2005) is a text-to-text generation technique which addresses exactly this problem. Sentence fusion systems take a set of related documents as input and output sentences "fused" from dependency structures of similar sentences. In this thesis we present a novel sentence fusion system which advances TS towards abstractive summarization by building a global representation of input sentences and generating a new sentence from this representation. The sentence fusion process includes two main tasks – dependency tree construction and dependency tree linearization, both of which we solve in a novel and effective way. Our tree construction method is largely unsupervised and generates grammatical sentences by taking syntactic and semantic knowledge into account without reliance on hand-crafted rules. Tree linearization is accomplished with a method that extends previous approaches but requires little overgeneration in comparison with them. Our method is also significantly more accurate than the previous ones because it utilizes features from several levels of linguistic organization (syntax, semantics, information structure). We test our system on a corpus of comparable biographies in German and obtain good readability results in an evaluation with native speakers. We also apply the same method to sentence compression (i.e., the task of producing a summary of a single sentence) in English and German and obtain results comparable to those reported by recent systems designed exclusively for this task.

Zusammenfassung

Die Popularität von Text-Zusammenfassung (TS) in der NLP-Gemeinschaft hat in den letzten Jahren stetig zugenommen. Dies ist aufgrund ihres praktischen Nutzens nicht verwunderlich: z. B. wäre automatische Textzusammenfassung mehrerer Dokumente sehr hilfreich angesichts der enormen Menge von Nachrichten, die täglich online erscheinen. Obwohl TS-Methoden sehr unterschiedlich sind, teilen die meisten von ihnen eine wichtige Eigenschaft: Sie sind extraktiv, und die am häufigsten benutzte Extraktionseinheit ist der Satz. Das heißt, dass die meisten TS-Systeme Zusammenfassungen aus extrahierten Sätzen bilden. Der extraktive Ansatz hat den bekannten Nachteil, dass sich Sätze aus verschiedenen Quellen überschneiden, aber auch gegenseitig ergänzen können. Dies hat zur Folge, dass extraktive Systeme oft nicht in der Lage sind, Zusammenfassungen zu generieren, die sowohl vollständig als auch nicht-redundant sind. Satzfusion (*sentence fusion*, Barzilay & McKeown (2005)) ist eine Text-to-Text Generierungstechnik, die genau dieses Problem angeht. Satzfusion ermöglicht es, aus den Dependenzstrukturen ähnlicher Sätze, die verwandten Dokumenten entstammen, neue Sätze zu generieren. In dieser Arbeit stellen wir ein neuartiges System vor, welches Satzfusion weiter in Richtung abstraktiver Textzusammenfassung entwickelt, indem erst eine globale Darstellung von Input-Sätzen aufgebaut wird und dann neue Sätze aus dieser Darstellung generiert werden. Im Wesentlichen beinhaltet Satzfusion zwei Aufgaben: Dependenzbaumkonstruktion und Dependenzbaumlinearisierung. Beide Aufgaben lösen wir auf eine neue und effiziente Art und Weise. Unsere Baumkonstruktionsmethode ist weitgehend unüberwacht und erzeugt grammatische Sätze, indem sie syntaktische und semantische Information berücksichtigt, ohne auf manuell geschriebene Regeln zurückzugreifen. Unsere Baumlinearisierungsmethode basiert auf bisherigen Ansätzen, ist aber im Vergleich mit ihnen deutlich effizienter. Überdies erreicht unsere Methode höhere Akkuratheit, da sie Wissen von verschiedenen Ebenen sprachlicher Analyse nutzt (Syntax, Semantik, Informationsstruktur). Wir testen unser System auf einem Korpus vergleichbarer Biographien in deutscher Sprache und erreichen gute Lesbarkeitsraten in einem Experiment mit Muttersprachlern. Wir übertragen dieselbe Methode auf Satzkomprimierung im Englischen und im Deutschen mit dem Ziel,

eine Satzzusamenfassung zu generieren, und erreichen Ergebnisse vergleichbar mit speziell fr diese Aufgabe entwickelten Systemen.

Acknowledgments

Michael Strube has been an excellent supervisor who *always has time* for his students. His guidance and interest in the work have helped shape this thesis, and I hope to have learned from his ability to focus on the right questions. Chatting with Michael (not only about work) has always been a pleasure, and many of his advices have already proven right.

I am indebted to Elke Teich who, being my distant supervisor, provided me with helpful comments at all the stages of the research and during the writing phase. Thanks to Elke, my *Promotionsvorgang* has been as easy, quick and smooth as it is in theory but as it is seldom in practice. I am also grateful to Mirella Lapata who immediately agreed to be the external reviewer of the thesis. I have enjoyed working with my colleagues at EML and learned a lot from them, especially from Christoph Müller, Margot Mieskes, Simone Ponzetto and Vivi Nastase. EML with its spirit of scientific endeavor is a beautiful and unique place which I feel sad to leave. I also appreciate the feedback I received from the Computational Linguistics Group at the University of Heidelberg. Apart from the people mentioned, I am grateful to all those from whom I learned and whose ideas I found inspiring throughout my life.

It is impossible to overestimate the support of my family, in particular my mother, my brother Andrei and my husband Armin whose encouragement has been with me from the very beginning. My mom's certainty that I shall write *a wonderful thesis*, my brother's stimulative inquiries from the first day (*When will you finally finish?!*) and Armin's insightful comments (which saved the thesis from many embarrassing mistakes) have also helped me to finish in a reasonable time.

Finally, I would like to acknowledge the financial support of the Klaus Tschira Foundation. A part of the comparable corpus I used comes from the Brockhaus Lexikon and is used with their permission.

Erklärung

Hiermit erkläre ich, die vorliegende Arbeit selbständig verfasst und keine anderen als die ausdrücklich genannten Quellen und Hilfsmittel verwendet zu haben.

(Katja Filippova)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis is about **sentence fusion** – a text-to-text generation technique which produces novel sentences from related documents. In this chapter we first introduce **text summarization** – an important natural language processing (NLP) application (Sec. 1.1), identify the shortcomings of existing summarization methods (Sec. 1.2) and show the potential of sentence fusion for text summarization (Sec. 1.3). Then we describe our contributions and outline the architecture of our sentence fusion system, called deFuser (Sec. 1.4), and finally provide an overview of the chapters of the thesis (Sec. 1.5).

1.1 Text Summarization

Text summarization (henceforth TS) concerns producing a **summary** of a single document or a set of documents. The former case is called **single-document** summarization, the latter one – **multi-document** summarization (henceforth MDS). A summary of a text is usually defined as follows:

> **A summary** "is a text that is produced from one or more texts, that contains a significant portion of the information from the original text(s), and that is no longer than half of the original text(s)" (Hovy, 2003, p. 584).

> **Text summarization**, in turn, is "the process of distilling the most important information from a source (or sources) to produce an abridged version for a particular user (or users) and task (or tasks)" (Mani & Maybury, 1999b, p. ix). It also gives the name to a sub-field of NLP which investigates ways of producing summaries automatically.

Since TS is a text-to-text application, it faces the challenges of text understanding as well as of text generation (Reiter & Dale, 2000). Text understanding is necessary to select important

content, i.e., for **content selection**, whereas the summary is the result of **summary generation** process.

The first TS systems were developed in the late 1950's (Luhn, 1958), and influential papers appeared further in the 1960s and 1970s (Edmundson, 1969; Skorochod'ko, 1972). However, it is only in the late 1990s and the beginning of this century that a strong interest in TS developed. The growing popularity of TS is reflected, e.g., in TS competitions (DUC and TAC) organized annually since 2001[1], a textbook about TS (Mani, 2001), an edited collection (Mani & Maybury, 1999a) and a special issue of the Computational Linguistics journal (Radev et al., 2002). Nowadays TS systems are designed to provide **generic** as well as **topic-** or **query-oriented** summaries. The former include all generally important points of a text whereas the latter include information determined to be important with respect to a query or a topic specified by the user. One also distinguishes between **indicative** and **informative** summaries. Summaries of the former kind only indicate what the input text is about whereas informative summaries can be used as substitute for the text.

Although TS methods vary considerably, most of them share one important property: they are **extractive**, and the most common **extraction unit** is the sentence – i.e., most TS systems build summaries from extracted sentences. The general approach employed by extractive methods is to rank sentences from a given set of related documents by their importance and then select the top scoring ones to fill a summary of a desired length. Most MDS systems check whether important sentences are different enough to avoid redundancy in the summary (Carbonell & Goldstein, 1998). Finally, some post-processing (sentence ordering, sentence compression or simplification) can be done in order to improve the coherence of the output text. Naturally, sentences pulled from different documents are unlikely to build a coherent text when combined together. This can be observed in the poor ratings of the linguistic quality in the DUC and TAC competitions.

Intuitively, the way humans summarize is very different from the extractive strategy, and indeed this has been confirmed in a series of psychological experiments (Kintsch & van Dijk, 1978). To be **abstractive** and more "human-like", an automatic TS system should **interpret** the input text, construct its (symbolic) representation, make necessary inferences and only then generate a summary from the representation (Spärck Jones, 1999).

Unfortunately, text interpretation on the level required for truly abstractive TS is not possible yet, and the attempts of doing abstractive TS have been limited to the use of domain-specific templates (Radev & McKeown, 1998). The consequence is that

> "[...] at present, if one is constructing a practical system, extraction seems more attractive." (Mani, 2001, p. 163)

[1] Document Understanding Conference (DUC) (`http://duc.nist.gov`) in the period 2001-2007; Text Analysis Conference (TAC) since 2008 (`http://www.nist.gov/tac`).

This citation is almost nine years old but, despite the steady interest and advances in TS, it seems that most progress has been achieved on the content selection and not on the generation side of TS. Thus, to date the absolute majority of the existing TS systems is purely extractive (Spärck-Jones, 2007).

1.2 Shortcomings of Extractive Summarization

Given the vast amount of information available on the Internet, it is not difficult to imagine scenarios where TS and especially MDS systems would be of a great help for the user. For example, consider online news. Nowadays, one can easily get hundreds of articles concerning the very same event, e.g., using Google News[2]. Clearly, these articles may overlap to a large extent. At the same time they might contain complementary or even contradictory information such as, e.g., the exact number of casualties in an airplane crash. It is unlikely that ordinary users read more than one news article about an event in order to get a more complete picture of what happened. In this scenario, a concise summary including complementary information from different sources might be preferred over a long list of similar news. The need for robust MDS algorithms in the news domain is reflected in the recent summarization tasks issued by DUC/TAC which all concern multi-document news summarization.

As an illustration of the extractive approach falling short, consider the four sentences in (1.1-1.4) which were manually extracted from four related news articles about the horrifying massacre in the South of Germany in March 2009:

(1.1) Several calls to tighten gun laws and monitor gun owners' accordance with storage requirement have been issued by politicians and other groups after 17-year-old Tim K., armed with a Beretta gun taken from his father's bedroom, killed 16 people in the small southwestern town of Winnenden, near Stuttgart.

(1.2) Kretschmer shot many of his victims in the head with his father's legally registered Beretta.

(1.3) Authorities say 17-year-old Tim Kretschmer used one of his father's weapons to gun down 15 people in a rampage that began at his former high school Wednesday.

(1.4) Kretschmer gunned down students and teachers at his former high school before fleeing on foot and by car, killing three more people, and eventually shooting himself in the head, police said.

[2] `http://news.google.com`

The four sentences above are clearly similar in that they are all about a high-school student, referred to as *Tim K., Kretschmer* or *Tim Kretschmer*, killing (*shooting, gunning down*) innocent people (*victims, students and teachers*) with a gun (*Beretta, weapon*). However, each of the sentences contains bits of information which other sentences lack. For example, it is only (1.1) which tells where the gun was taken from or that the shooting took place near Stuttgart; likewise, it is only (1.2) which points out to the fact that the gun was legally registered and only (1.3) says that the massacre took place on Wednesday; (1.4) tells us who the victims were.

Suppose that an extractive MDS system is fed the news about the massacre and is required to produce a short summary. It is highly likely that the four sentences above would get a very high rank as each of them can be viewed as a summary of the event. Now, an extractive TS system faces the following problem: either it includes only one sentence in the summary, but then some information gets lost, or it selects two or more sentences and thus makes the summary redundant. This example illustrates the **trade-off between non-redundancy and completeness** which is typical of extractive systems.

Another point worth mentioning here is that some sentences – actually, all but (1.2) – contain information irrelevant for the summary. For example, the main clause of (1.1) (from *several* to *after*) could be eliminated altogether given that the information about the massacre and not the impact it made on the laws is of interest. Similarly, the attribution of (1.3) to the authorities and (1.4) to the police could be omitted, given the value of space. This motivates **sentence compression** – a technique of shortening the sentence while retaining its main points as well as its grammaticality. Sentence compression has attracted considerable attention in the last decade and many different methods have been developed (Grefenstette, 1998; Knight & Marcu, 2000; Jing, 2001; Riezler et al., 2003; McDonald, 2006; Clarke & Lapata, 2008, inter alia). However, it has also been reported that extraction, even when combined with compression, leads to suboptimal results (Daumé III & Marcu, 2002). An overview of research in sentence compression is given in Chapter 8.

If extracted sentences tend to miss interesting information and at the same time include irrelevant information, what would a better summary sentence look like? Arguably, given that the space is very limited (e.g., the summary is to appear as a snippet), (1.5) would be a good choice:

(1.5) Tim K. killed 16 people with his father's gun.

Given that there is enough space available, a more complete summary would be appropriate. For example, (1.6) includes pieces of information from all the four sentences:

(1.6) 17-years-old Tim Kretschmer, armed with his father's legally registered gun, killed students and teachers at his former high-school in the small town of Winnenden.

Thus, it would be of great use if there existed a way of generating novel sentences from text such that these sentences incorporate important content from several sources and exclude irrelevant information. But how could this be done if deep text interpretation and hence abstractive TS are not possible?

Several (although not many) approaches to generating novel sentences directly from text have been suggested. They differ in aims as well as in the depth of linguistic analysis they require. For example, the tasks of headline and table-of-contents generation have been explored with relatively shallow features, e.g., PoS tags (Banko et al., 2000), *tf.idf* (Jin & Hauptmann, 2003), or bigrams (Branavan et al., 2007). Unfortunately, headlines as well as chapter and section titles are very different from "normal" declarative sentences encountered, e.g., in the news. The latter are much longer and have a complete grammatical structure whereas the former are often a few words long and constitute a single noun phrase. Therefore, these methods cannot be applied to summary sentence generation. Wan et al. (2005, 2009) introduce methods which are of a more direct relevance to summarization because they are capable of generating complete natural language sentences. Given a set of important words extracted from a single document (Wan et al., 2008), Wan et al. (2009) find the best dependency tree covering those words as well as the best word order. The problem with this approach is that the generated structure may have a meaning quite different from and even contrary to what is implied in the input. The method generates a sentence whose likelihood is maximized with respect to a corpus and not to the text these words were extracted from. For example, given the words *John*, *Mary* and *loves* extracted from a text including the sentence *Mary loves John*, a sentence with a totally different meaning – *John loves Mary* – may be generated. Since the goal of TS is to convey important information, consistency with the input is a crucial issue.

1.3 Sentence Fusion by Barzilay & McKeown (2005)

Unlike the before mentioned approaches, **sentence fusion** – a technique introduced by Barzilay & McKeown (2005) – provides a more compelling answer to the question of how novel grammatical sentences can be generated from text.

> **Sentence fusion** is a "text-to-text generation technique which, given a set of similar sentences, produces a new sentence containing the information common to most sentences in the set" (Barzilay & McKeown, 2005, p. 298).

The algorithm of Barzilay & McKeown (2005) is designed for generic MDS in the news domain. It takes a set of related news as input, clusters similar sentences and generates, or fuses, a novel sentence from the dependency trees of similar sentences which conveys the content shared among the sentences from the cluster. Importantly, Barzilay & McKeown

(2005) solve two very different and difficult problems of TS simultaneously – those concerning content selection and generation. By definition of the summary, fused sentences should convey the important part of the information from the source sentences; they must also be grammatical and make sense to the reader. Given that it is the gist of an event which is repeated in different news articles, extraction of common part is a reasonable approximation of importance. At the same time, it is natural to expect that the shared part includes the grammatical skeleton of the event – the verb with its obligatory arguments. It should be emphasized here that redundancy in the input, typical of MDS, is an important requirement for the method. In the context of single-document summarization, where redundancy is missing from the input, generic sentence fusion has been shown to be an ill-defined task (Daumé III & Marcu, 2004).

The described kind of fusion in which only the **shared part** of the input content is extracted was later termed **intersection fusion** by Krahmer et al. (2008) as opposed to **union fusion**, which combines complementary information from different sources in a single sentence. Given a group of similar sentences in (1.1-1.4), (1.5) and (1.6) may serve as examples of intersection and union fusion, respectively. Similar to Krahmer et al. (2008), we do not restrict fusion to intersection only and for the present purposes define it as follows:

> **Sentence fusion** is a text-to-text generation technique which, given a set of similar sentences, produces a new sentence conveying all or a portion of the relevant information from the input.

Defined in this way sentence fusion holds promise to be useful not only for generic but also for query-oriented TS. It is no longer restricted to intersection but also includes union fusion and anything in between, depending on the needs. It is this broader definition of fusion which we are going to use henceforth. Importantly, sentence fusion can be viewed as a middle-ground between extractive and abstractive summarization. In essence, it is still an extractive method (Spärck-Jones, 2007) but with finer granularity – unlike previous approaches, its extraction unit is not the sentence but the **syntactic dependency**. This finer granularity opens new possibilities for TS and is an important step towards abstractive summarization as it allows to generate unseen sentences (i.e., sentences not present in the input).

The original sentence fusion algorithm of Barzilay & McKeown (2005) is presented in detail in Chapter 7, and here we give a higher-level overview of the method. In a nutshell, it proceeds as follows:

1. Groups of similar sentences, such as (1.1-1.4), are built from a set of related news articles. Every group serves as input to the fusion system which generates one new sentence per group.

2. The dependency trees of input sentences, i.e., the sentences from one group, are compared and the one which shares most structural similarities to other trees in the group

is selected. Tree similarity is computed during pairwise tree alignment when identical nodes or paraphrases as well as dependency edges are aligned. The selected tree, which is called **the basis tree**, is then modified in two respects:

- First, alternative paths and subtrees are inserted under the condition that they appear often enough in the cluster. This is process is called **tree augmentation**.
- Second, some of the subtrees are removed given that they are not grammatically obligatory and do not appear in many input trees. This is called **tree pruning**. The set of grammatically optional arguments is predefined and includes prepositional phrases (PPs), adverbs and certain clauses.

3. The resulting dependency structure, which is not necessarily a tree, is **linearized** by overgeneration and ranking. That is, all possible strings are generated from the structure and then ranked with a trigram language model. The variant with the lowest entropy is selected and output as the result of sentence fusion. For (1.1-1.4) an appropriate result would be (1.5).

Albeit elegant, the described approach has some important deficiencies which motivate the development of an alternative approach to sentence fusion:

Scope of fusion. This point has been mentioned before, here we elaborate it further. The essence of the approach of Barzilay & McKeown (2005) is to deliver the shared content of the input by modifying one of the input structures – the basis tree – with a few quite restrictive augmentation rules. As a consequence, sentences different from the basis one do not contribute any content to the output at all. This is a severe limitation because in many situations more intensive union fusion is of a greater use (Krahmer et al., 2008). Turning back to the massacre example, (1.6) cannot be produced with the original sentence fusion approach although it would be more appropriate than (1.5) in many scenarios. Therefore, it would be useful if the generation of a new sentence would not be biased to one sentence but would rely on all the content available in the input.

Grammaticality. As we have noted in the beginning of this section, to some extent grammaticality is ensured by identification of the structure shared among the input trees. Also, since a novel sentence appears as a result of basis tree modification, it is expected to have enough similarities with the basis tree which is per default grammatical. However, there appears to be a trade-off between how novel and how grammatical the fused sentence is. On the one hand, it is desirable to generate output which is different from each of the input sentences – this is what we have argued relating to the scope of fusion. On the other hand, more

"intensive" fusion seems to affect the grammaticality of the output: less restricted augmentation is likely to produce ungrammatical structures. Given that their goal is intersection fusion, Barzilay & McKeown (2005) follow the safer path of highly restrictive augmentation in order to minimize the chances of generating ungrammatical sentences. For example, one of the augmentation rules states that a new path or node can be inserted given that it appears in at least half of the input sentences, which is quite a high threshold. Possible extensions to the approach of Barzilay & McKeown (2005) have been proposed. Marsi & Krahmer (2007) and Krahmer et al. (2008) describe an architecture where one may choose between intersection and union fusion depending on the needs of TS. Yet they do not report any evaluation results and it is unclear how well such approaches would perform in practice. It is foreseeable that more intensive fusion would require more tree augmentation and pruning rules. Even in the present configuration some rules are clearly too general to hold universally – e.g., *PPs can be pruned*. Writing more rules would require more human labor and would hinder portability of the method to other languages and domains. Apart from that, rule-based systems are often difficult to maintain and require heuristics to resolve conflicting rules.

The second problem with grammaticality is unrelated to the scope of fusion and arises during linearization. In fact, Barzilay & McKeown (2005) partially shift the burden of choosing a grammatical structure till linearization. However, word order generation is itself a difficult problem and a simple trigram model is not sufficient to gauge sentence grammaticality because it cannot take into account long-distance dependencies (Chapter 6 provides more detail). Thus, an unacceptable string can be produced even from a grammatical structure.

Portability. Nowadays the questions of domain independence and method portability become more and more important. For example, one of the appeals of statistical machine translation (SMT) is that SMT systems can be ported to other languages and domains provided that there is enough parallel data available. This also explains the interest in unsupervised methods. Barzilay & McKeown (2005) do not explicitly utilize language-specific knowledge. However, the linearization method is expected to work better on languages with relatively rigid word order. Although the augmentation and pruning rules are general and seem to hold across different languages, their refinement is likely to make them language-dependent (e.g., subjects are obligatory in English but are optional in Spanish or in Slavic languages).

1.4 Contributions of this Thesis

This thesis presents a novel sentence fusion system which, similarly to the original one by Barzilay & McKeown (2005), operates on dependency structures. The general architecture is hardly different: the system gets a set of related documents as input, extracts groups of similar

sentences from them, builds a new dependency tree for each group and finally linearizes it, i.e., converts this tree into a sentence. These are exactly the three steps described in the previous section. However, each of the three steps is accomplished quite differently in the present approach. The main differences concern the second and the third phases and are motivated by the following considerations which address the issues discussed at the end of the previous section:

Scope of fusion. One of the goals of our research is to develop a method which would go beyond intersection fusion. We have shown that union fusion is more appropriate in many cases. Similarly, topic-oriented fusion requires methods which generate novel sentences covering not the most frequent points from the input but the ones relevant with respect to a given topic. In this case, approaches biased to one input tree are of little use, and a complete representation of all the input content is needed.

Grammaticality. Another goal is to find a way of generating grammatical sentences without the severe restrictions of the original fusion method. To a large extent grammaticality depends on the presence of obligatory arguments such as the subject or the direct object for the finite verb, determiners for nouns, etc. This is a limited view on grammaticality but this perspective is usually adopted in NLP applications. The rules of Barzilay & McKeown (2005) are also aimed to ensure grammaticality by retaining obligatory arguments and pruning optional ones. As we have pointed out earlier, the problem with the rule-based approaches is that they are expensive and require either human labor or specific resources which are not readily available for most languages. Hence, one of our goals is to find a way of ensuring grammaticality without adhering to complex rules or expensive resources.

Grammatical well-formedness is one of the facets of utterance acceptability, the other one being semantic soundness. The approach of Barzilay & McKeown (2005) does not implement any semantic rules during tree modification but utilizes lexical information during alignment. In our system we want to explore an alternative approach and prove semantic soundness of the structure we are building. For example, given (1.1-1.4) as input, we want to make sure that syntactically well-formed but semantically unsound sentences such as (1.7-1.8) are not generated.

(1.7) *??*Tim K. shot teachers and people with his father's gun.

(1.8) *??*Tim K. killed students and victims with a Beretta gun.

Finally, to minimize the amount of errors during linearization and to avoid inefficient overgeneration, we design a method which can cope with long-distance dependencies and does not need to consider all possible strings to find the best one.

Portability. Yet another goal of our research is to make the method portable to other languages. Therefore, we want to minimize the use of hand-crafted rules and resources unavailable for most languages. Actually, that sentence fusion relies on dependency structures is already a strong requirement.

The three points listed above – grammaticality, scope of fusion, portability – were taken into account in the design of **deFuser** – a sentence fusion system developed for German (hence **de** in the name) which generates novel sentences from a set of biographies about a person. The architecture of deFuser is presented in Figure 1.1. deFuser consists of five modules:

1. **Sentence grouper** takes related documents as input and outputs groups of similar sentences. In Figure 1.1, the first group includes three, the second one two and the last one four sentences. Again, sentences in (1.1-1.4) are very likely to be grouped together.

2. **Tree transformer** gets a group of similar sentences as input, all parsed, and transforms their dependency trees. The main goal of the transformations is to make the structure more semantically motivated. Some trees become dependency graphs as a result ('1a' and '1b' in Fig. 1.1).

3. **Aligner/merger** operates on transformed trees and builds a complete representation of the content of the input sentences. This representation is to a large extent syntactic but also covers many semantic relations. This is an important extension to the original sentence fusion method which never abstracts from single trees to a **global** representation. This step brings our approach closer to abstractive systems which generate novel sentences from a complete representation of the input. The output of this module is a graph covering the content of all the sentences. Going back to the similar sentences in (1.1-1.4), given that an accurate dependency parser and a lexical resource such as WordNet are available, one can get a complete graph covering the content of all the four sentences, such as the one in Figure 1.2[3].

4. **Graph compressor** generates a novel dependency tree by compressing the complete graph. For example, the graph in Figure 1.2 can be compressed to the dependency tree of the "intersection" sentence in (1.5) as well as to the one of (1.6) – the nodes and edges to be retained are highlighted in green in Figures 1.3a and 1.3b respectively. Concerning this module, the important improvements here are as follows:

    - Graph compressor considers **all** the information from the input globally and thus deals with a complete representation and not with single trees.

[3]This and the two following graphs were generated with aiSee: `http://www.aisee.com`.

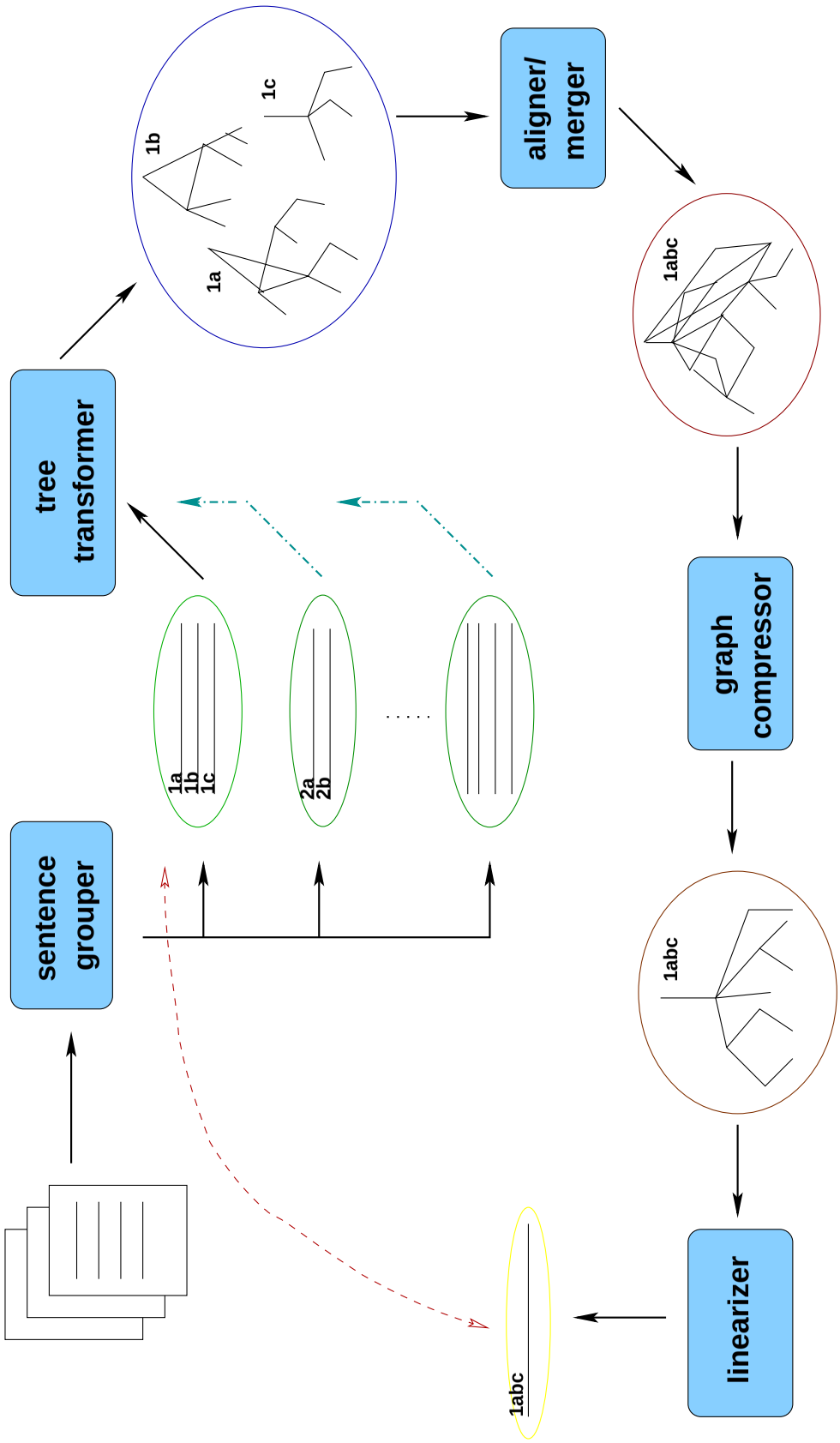


Figure 1.1: deFuser system overview

- It outputs a **single** dependency tree and not a dependency structure. Thus, the burden of selecting the grammatical structure is never shifted to the linearization module.
- It takes not only syntactic but also **semantic** knowledge into account and relies neither on hand-crafted rules, nor on a grammar resource.

5. **Linearizer** converts the tree into a sentence by taking several linguistic factors into account. It is also significantly more efficient than the linearization technique adopted in other fusion and generation systems because it does not need to consider all possibilities to find the best one. The resulting sentence summarizes the content of exactly one group (see the red line from the group of '1a,1b,1c' to the output sentence).

One may also notice that sentence compression, which we mentioned on page 4, can be viewed as a kind of trivial fusion when nothing is fused but when certain elements are eliminated. Indeed, if the task of a fusion system is to take a set of related sentences and produce a novel one which would retain the important information and be grammatical, then the compression system does exactly the same for a single sentence. From this it follows that feeding a single sentence into a fusion system could be a test which checks, e.g., how well the system treats grammaticality. If the system fails to produce readable output from one sentence, it is highly unlikely that it will perform better when several sentences are provided. Thus, one of our aspirations is to demonstrate the applicability of deFuser to sentence compression.

To summarize, the contributions of this thesis are as follows:

1. A novel sentence fusion technique is presented which advances TS one step further towards abstraction compared with previous methods.
2. Grammaticality of fused sentences is ensured without reliance on manually crafted rules or expensive resources.
3. Grammaticality is enforced during all the stages of the generation process with syntactic and semantic constraints.
4. The method is largely unsupervised and its independence from language-specific resources makes it portable to other languages.
5. As far as we are aware, deFuser is the first sentence fusion and compression system for German.
6. deFuser achieves good readability results in fusion and compression on German data.
7. Its performance in a sentence compression experiment on English data is comparable with the best of the existing systems.

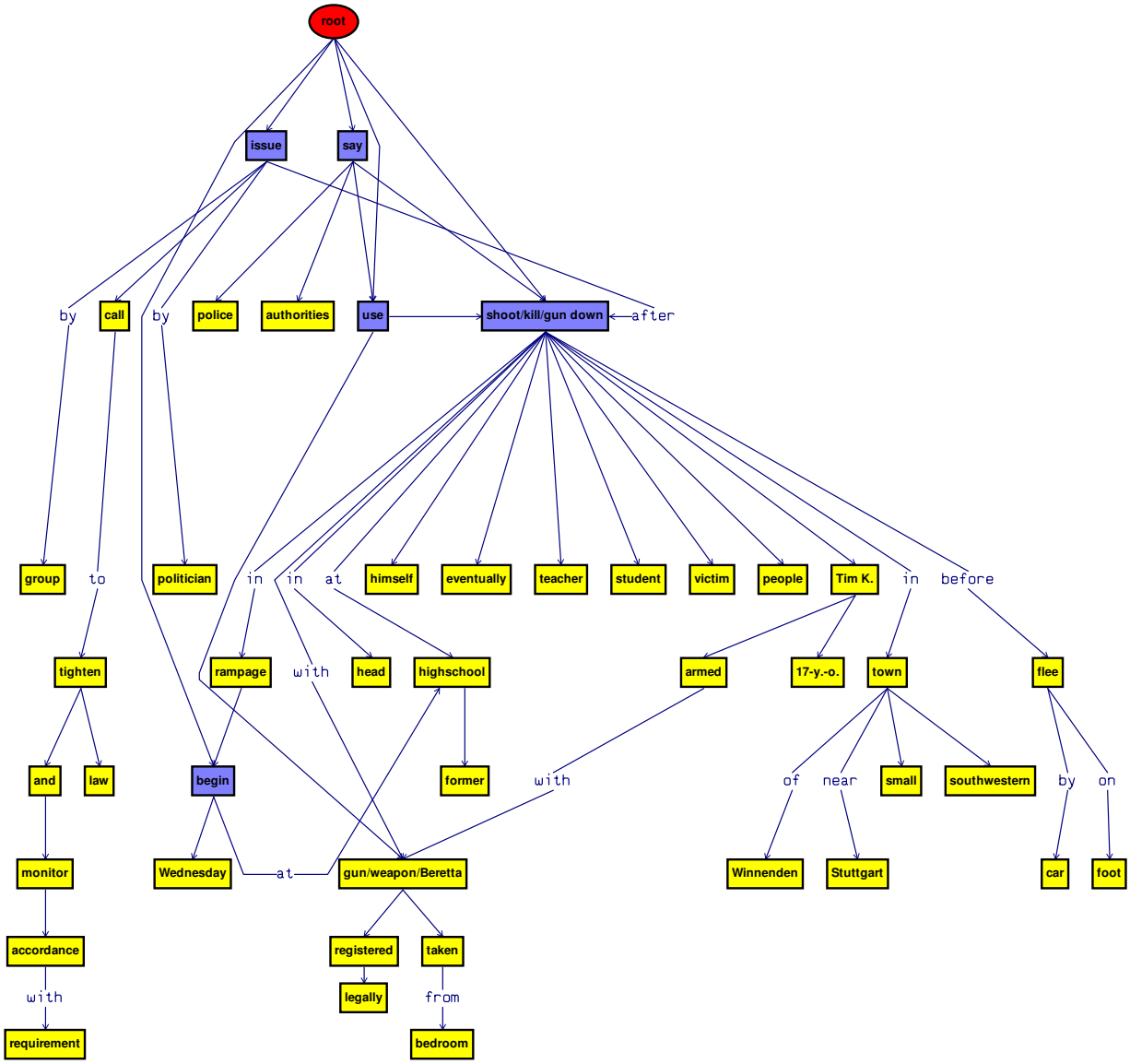
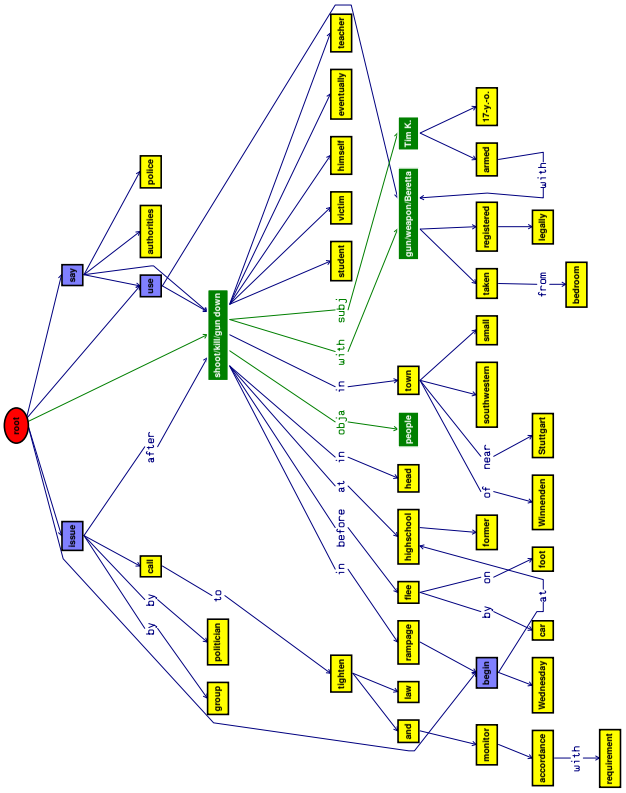
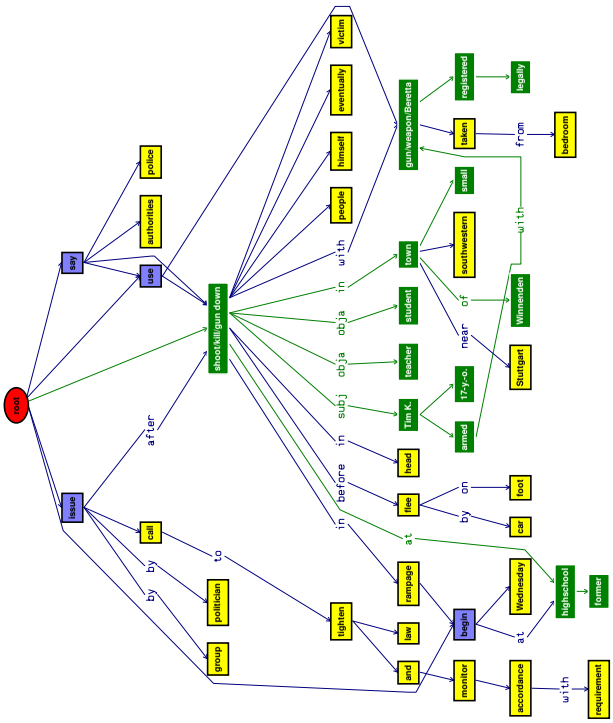


Figure 1.2: Graph covering the content of four sentences in (1.1-1.4)



(a) The dependency tree of (1.5)



(b) The dependency tree of (1.6)

Figure 1.3: The trees corresponding to (1.5-1.6) highlighted in the graph

8. The formalism adopted in this work can be easily extended with further constraints concerning content selection as well as summary generation.

1.5 Thesis Overview

- Chapter 2 describes the corpora and annotation used throughout the thesis. The information about all the data sets we used is packed in one chapter for convenience. Throughout the thesis references to different corpora are made, so the reader might find it handy to refer to this chapter for clarifications.
- Chapter 3 introduces the sentence grouping module which extracts related sentences from similar documents and clusters them (see sentence grouper in Fig. 1.1).
- Chapter 4 presents our method for generating new dependency trees from a set of related sentences. It describes the details of the tree transformer, aligner/merger and graph compressor (see Fig. 1.1).
- Chapter 5 lays the linguistic foundation for our tree linearization method and presents empirical results of a corpus study, of an experiment with native speakers, and a small generation experiment.
- Chapter 6 presents the tree linearization method (see the 'linearizer' box in Fig. 1.1) which we initially developed for German and then adapted for English. The results of the evaluation experiments are also reported there.
- Chapter 7 concerns evaluation and compares deFuser with a reimplementation of the method of Barzilay & McKeown (2005). A discussion and error analysis can also be found in this chapter.
- Chapter 8 demonstrates how deFuser can be applied to sentence compression and reports the results of experiments on English and German data.
- Chapter 9 concludes our work and outlines the directions for future research.

Guide for the reader: Of course, some parts of this thesis are more interesting than others. This guide is here for readers willing to know what the main ideas of the thesis are, so that they can proceed to them right away. Chapters 3, 4 and 6 present what constitutes deFuser. The sentence grouping algorithm is relatively straightforward and does not represent a major contribution of the thesis. Chapters 4 and 6 constitute the core of the thesis and each addresses the grammaticality point in a novel way. The novelty of the tree generation part concerns

working with a global representation (Sec. 4.3) and a cheap way of getting a grammatical dependency tree from this representation (Sec. 4.4). Another novel point is the way of integrating semantic knowledge into the system (Sec. 4.4.3.4) which in earlier work has been used during tree alignment only. The main claim of tree linearization part is that clause constituents in German possess certain weights which can be used to put them in a right order (Sec. 6.4). These weights can be estimated from such properties as the syntactic function, semantic class, length in words, etc. Separating clause constituents ordering from ordering words within them is justified because the latter task is much easier and can be solved accurately with a trigram language model (Sec. 6.5). Chapter 5 is for more linguistics-oriented readers interested in local coherence and information structure. The tree compression system in Chapter 8 is not a separate contribution. It can be seen as a one-sentence fusion system whose core is basically the same as that of deFuser.

1.6 Generated Resources and Published Work

Most parts of this thesis have been published earlier. The graph compression method (Chapter 4) was described in Filippova & Strube (2008b). The linguistic underpinning of the linearization method (Chapter 5) was first presented in Filippova & Strube (2007b). The constituent ordering method – an important part of the linearization algorithm (Chapter 6) – was introduced for German in Filippova & Strube (2007a). The combined linearization method was presented in Filippova & Strube (2009). Finally, the results of applying deFuser to sentence compression (Chapter 8) were reported in Filippova & Strube (2008a).

The WikiBiography corpus and the corpus of comparable biographies, CoCoBi, are available for download from `http://www.eml-research.de/~filippova`.

Chapter 2

Data and Annotation

The sentence fusion and compression algorithm presented in this thesis was initially developed for and tested on German data. The sentence compression part was further adapted for English. This chapter introduces the corpora we used for training and evaluation for both tasks on both languages. Sections 2.1 and 2.2 are about the German and English corpora, respectively. Section 2.3 presents a short discussion concerning the annotation differences between the corpora.

2.1 German Corpora

In this section we present a corpus of comparable biographies used in our sentence fusion experiments (Sec. 2.1.1). We also describe a larger corpus from which statistics necessary for deFuser are calculated (Sec. 2.1.2) and a corpus of news used in our sentence compression experiments (Sec. 2.1.3).

2.1.1 CoCoBi

Sentence fusion is applied to sets of similar sentences and therefore a corpus of related documents is required to test a fusion system. Such corpora are called **comparable**, similar to parallel data for machine translation. Examples of comparable corpora used in NLP include the data issued by DUC/TAC, gospels from the Bible (Nelken & Shieber, 2006), or articles from Encyclopedia Britannica (Barzilay & Elhadad, 2003). For development and testing of our fusion method we prepared a *co*rpus of *co*mparable *bi*ographies in German, called CoCoBi. This corpus is a collection of about 400 biographies gathered from the Internet[1]. These

[1] `http://de.wikipedia.org`,
`http://home.datacomm.ch/biografien`,
`http://biographie.net/de`,

biographies describe 140 different people, and the number of articles for one person ranges from two to four, being three on average. Despite obvious similarities between articles about one person, neither identical content nor identical ordering of information can be expected.

We decided to use a corpus of biographies for the following reasons:

1. Biography summarization is an existing NLP application (Mani, 2001).
2. Biographies are rich in events and different sources often provide complementary information about the same event (e.g., location and time).
3. Arguably, it is easier to identify similar sentences in biographies than in texts of other genres. For example, numerous dates and locations provide a good indication of sentence similarity.

Annotation Pipeline. CoCoBi is automatically preprocessed. The preprocessing pipeline comprises the following steps:

- Sentence boundaries are identified with a Perl CPAN module[2] which utilizes a large set of common abbreviations.
- The sentences are split into tokens using simple heuristics.
- The TnT tagger (Brants, 2000) is used for part of speech tagging.
- TreeTagger (Schmid, 1997) is used for lemmatization.
- The sentences are parsed with the Weighted Constraint Dependency Grammar (WCDG) parser which shows the state-of-the-art results on German data (Foth & Menzel, 2006). An important drawback of the parser is that it is considerably slow. As a result of the limit of 30 minutes per sentence we set, some of the sentences are left unparsed. Dependency parsers have been reported to be more accurate on German data than phrase-structure ones. Kübler & Prokic (2006) attribute this to better treatment of coordinated constructions and long-distance dependencies by dependency parsers.
- References to the biographee – pronominal and proper (first, last) names – are identified automatically. This partial coreference resolution can be done easily given the biography genre. Our simple pronoun resolution rule states that every personal pronoun which agrees with the biographee in number and gender refers to the person. This rule held in practically all cases we checked manually on a subset of CoCoBi. Other kinds of

`http://www.weltchronik.de/ws/bio/main.htm`,
`http://www.brockhaus-suche.de/suche`

[2] `http://search.cpan.org/~holsten/Lingua-DE-Sentence-0.07/Sentence.pm`

references (i.e., neither by name nor by pronoun) such as *der berühmte Physiker* (*the famous physicist*) are not resolved.

- Discourse connectives – e.g., *denn* (*because*), *außerdem* (*apart from that*) – are identified with an extensive list (about 200 connectives) which was made available[3] by Institut für Deutsche Sprache (Institute for German Language, IDS), Mannheim, Germany.

- Temporal expressions are identified with a few rules. The annotation distinguishes between absolute and relative expressions. *Im Jahr 1890* (*in the year 1890*) is an example of an absolute temporal expression; *im selben Jahr* (*in the same year*) or *danach* (*after that, later*) are examples of relative temporal expressions.

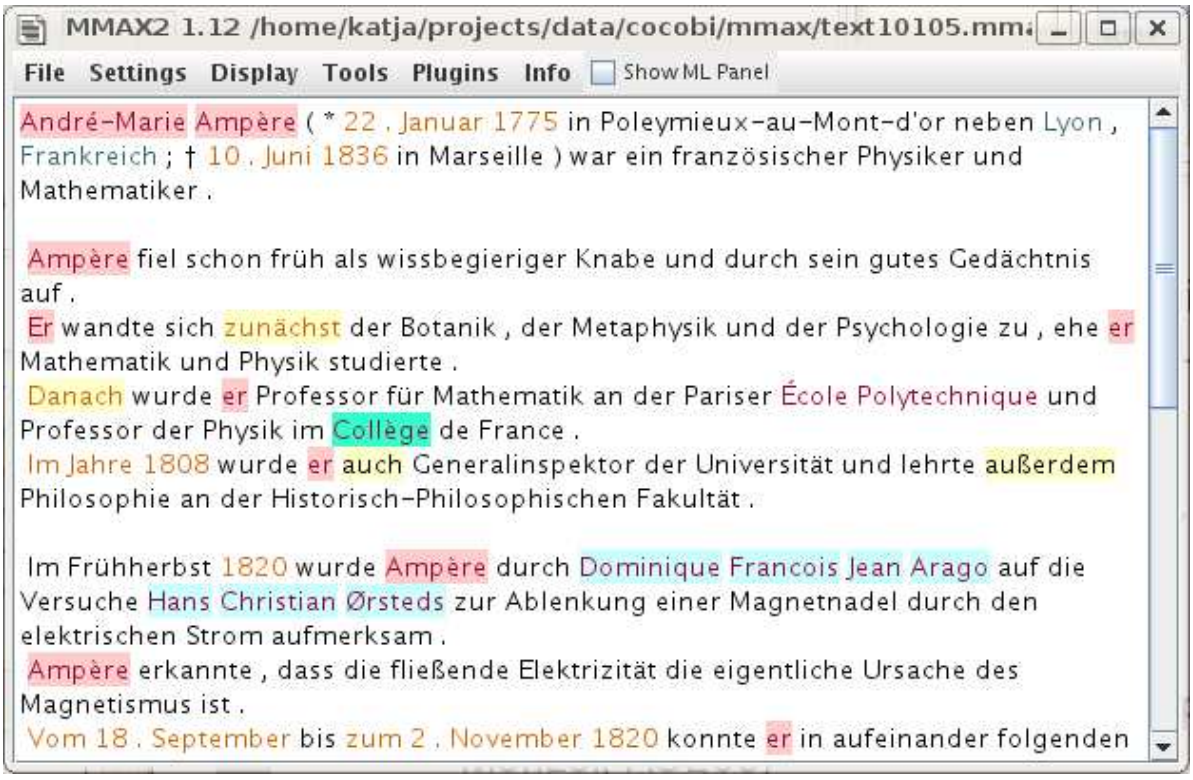
- Named entities (NE) recognized by the tagger are classified as *loc*ation, *person*, *org*anization with a large lexicon. The tag *unknown* is assigned in cases when the NE is not found. Initially, the lexicon contained the list of people for whom we collected the biographies, places listed in the German Wikipedia under some "locational" categories (e.g., STADT IN EUROPA – *city in Europe*) and the locations found in the first sentence of almost every biography (see Fig. 2.1a). We further enriched the list of people by quering every unclassified NE in Wikipedia and checking whether there is a corresponding article and whether it belongs to the categories MANN (*man*) or WOMAN (*woman*). The lexicon was further enriched with a sequence of iterations through the data when NEs found in a coordinated construction with some annotated NE were classified. For example, given the prepositional phrase *in Bourg und Lyon* (*in Bourg and Lyon*) with the NE *Lyon* classified as *loc* and the NE *Bourg* unclassified, we annotate *Bourg* as a location and then add it to the lexicon. In total the lexicon contains about 9,000 classified entries.

Figure 2.1 shows screenshots of two biographies of André Marie Ampère as displayed in the MMAX2 annotation tool[4] (Müller & Strube, 2006). Annotated references to the biographee (here, Ampère) are highlighted with red; references to other people (*Dominique Francois Jean Arago*) are highlighted with blue. Orange font and green fonts are used for temporal expressions resp. locations. Unclassified NEs are highlighted with green background. *École polythechnique* is recognized as organization and is displayed with red font. Yellow background is used for discourse connectives (*zunächst* (*first*)).

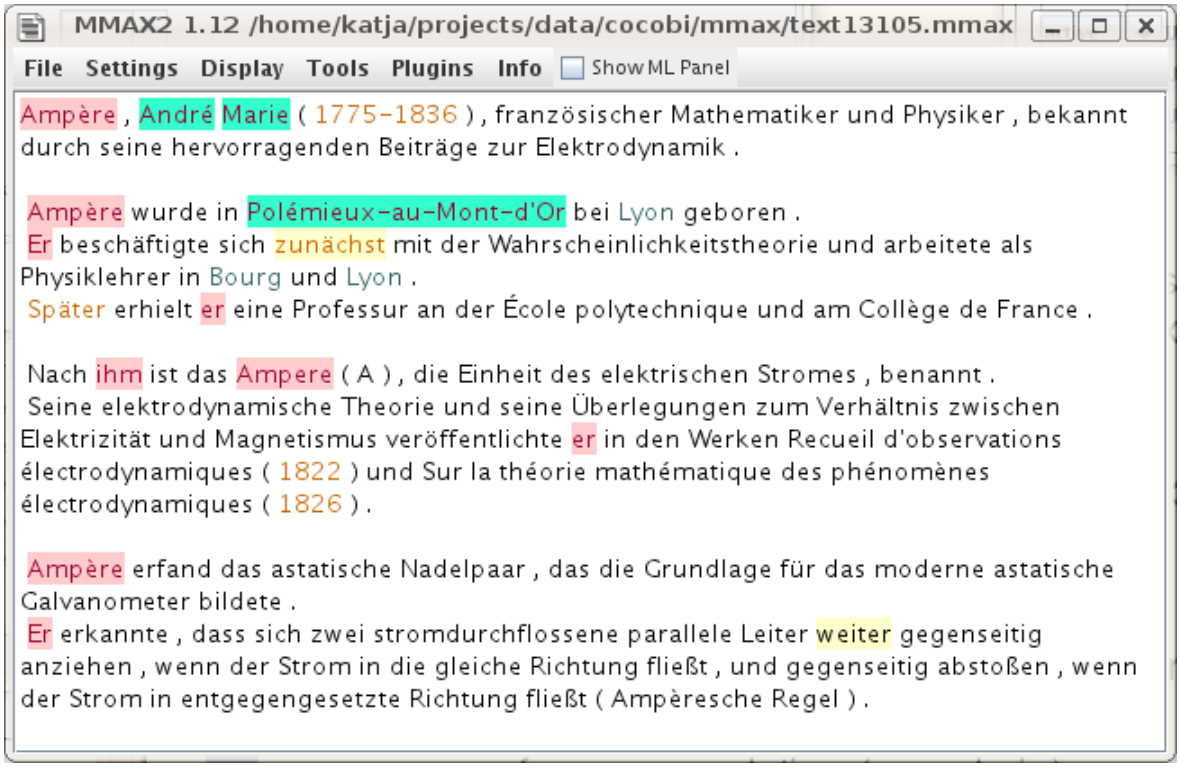
Table 2.1 gives the size of CoCoBi and other German corpora in tokens, sentences and single documents.

[3] `http://hypermedia.ids-mannheim.de/pls/public/gramwb.ansicht`

[4] Available for download from `http://mmax2.sourceforge.net`.



(a) An annotated biography of Ampère from Wikipedia



(b) An annotated biography of Ampère from Brockhaus Lexikon

Figure 2.1: Screenshots of annotated data

	tokens	sentences	articles
CoCoBi	221,571	9,844	400
WikiBiography	1,119,341	52,680	3,224
TüBa-D/Z	364,046	20,052	1,000

Table 2.1: Size of German corpora in words, sentences, articles

2.1.2 WikiBiography

In recent years Wikipedia[5] has become a valuable semantic resource for many NLP applications mainly because of its coverage and steadily growing size[6], the category information it provides (Ponzetto & Strube, 2007a), the rich link structure (Milne & Witten, 2008) and extensive information on practically every concept (Gabrilovich & Markovitch, 2007). Apart from being a considerable source of world and semantic knowledge, Wikipedia itself is a huge corpus of clean, well-maintained articles ready to be used. We build a corpus of biographies extracted from the German Wikipedia in 2006-2007 which includes about 3,200 articles. These are automatically annotated the same way as CoCoBi. A part of WikiBiography consisting of 1,200 biographies is available for download[7]. All the biographies in CoCoBi which come from Wikipedia are also included in WikiBiography. The tree linearization algorithm (Chapter 6) is evaluated on a part of the WikiBiography corpus.

2.1.3 TüBa-D/Z

Another German corpus we use is TüBa-D/Z (Telljohann et al., 2003)[8] – a collection of 1,000 newspaper articles which appeared in the end of the 1990s in *Die Tageszeitung*[9]. Sentence boundaries, morphology, dependency structure and anaphoric relations are manually annotated in this corpus. The annotation is converted into the same dependency format as the one that the WCDG parser produces (Versley, 2005).

We use this corpus to carry out sentence compression experiments. To make a justified comparison of the results across the two languages, we select a German corpus of the same genre as the English corpus, i.e., a corpus of news (see the section below about the English counterpart).

[5] `http://www.wikipedia.org`
[6] See `http://en.wikipedia.org/wiki/Wikipedia:Modelling_Wikipedia's_growth`.
[7] Available from `http://www.eml-research.de/nlp/download/wikibiography`.
[8] The corpus is available from `http://www.sfs.uni-tuebingen.de/en_tuebadz.shtml`.
[9] `http://www.taz.de`

	tokens	sentences	articles
compr. corpus	76,705	3,176	82
WSJ	19,503,448	787,782	46,448

Table 2.2: Size of English corpora in tokens, sentences, articles

2.2 English Corpora

In this section we describe the English corpora used to test the performance of deFuser on the task of sentence compression (Sec. 2.2.1) and to collect statistics required by our method (Sec. 2.2.2).

2.2.1 Compression Corpus

To evaluate the performance of deFuser on sentence compression we use the freely available corpus of compressed news in English[10], distributed by the University of Edinburgh. It is a document-based compression collection from the British National Corpus and American News Text Corpus which consists of 82 news stories. We parse the corpus with RASP (Briscoe et al., 2006) and with the Stanford PCFG parser (Klein & Manning, 2003). The output of RASP is a set of dependency relations whereas the Stanford parser provides an option for converting the output into dependency format (de Marneffe et al., 2006).

RASP has been used by Clarke & Lapata (2008) whose sentence compression results we compare with ours (see Chapter 8). We use not only RASP but also the Stanford parser for several reasons. First, a comparison between the Stanford parser and two other dependency parsers, MiniPar and Link Parser (Sleator & Temperley, 1993), showed a decent performance of the former (de Marneffe et al., 2006). Apart from being accurate, the Stanford parser has an elaborated set of dependency relations (55 vs. 15 of RASP) which is not overly large. The size of the relation set is important for deFuser as we will show in Chapter 4. It is also of interest to see to what extent the choice of the parser influences the performance.

Information on the size of the compression corpus in tokens, sentences and single documents is provided in Table 2.2.

2.2.2 WSJ Corpus

We take a subset of the TIPSTER[11] corpus – all Wall Street Journal articles from the year 1987 – and automatically annotate it with sentence boundaries, part of speech tags and depen-

[10] The corpus is available from `http://homepages.inf.ed.ac.uk/s0460084/data`.

[11] See `http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC93T3A`.

dency relations using the Stanford parser[12]. The size of the corpus in tokens, sentences and documents is given in Table 2.2. The tree linearization algorithm (Chapter 6) is evaluated on a portion of the WSJ corpus.

2.3 Discussion

Although all the corpora are annotated with dependency relations, there are a few differences between the annotations of the English and German data sets. The phrase to dependency structure conversion done by the Stanford parser makes the semantic head of a clause its syntactic head per default. For example, in the sentence *'He is right'* it is the adjective *right* which is the root of the tree, the verb *is* attached to *right* with the *cop*ula label. Unlike that, sentences from the German corpora always have a finite verb as the root. To unify the formats, we slightly modified the source code of the Stanford parser to make the verb the root of the tree in all cases.

The dependency sets also differ. The German dependency set contains 34 dependency types while the English one has 55 (see Tables 2.3 and 2.4[13]). However, some of the English labels are very general and assigned only when the exact label cannot be recovered from the prase structure parse. Such general labels are marked with * in Table 2.4: e.g., *dep* stands for dependency in general. Of course, some labels are considerably more frequent than others. For example, *obja2* assigned by the WCDG parser (see Table 2.3) is encountered with verbs which have two accusative objects in their subcategorization frame, such as *lehren* (*to teach*).

2.4 Summary

The section has presented the data and annotation which served as input to deFuser. All the data is annotated with sentence boundaries, parts of speech and syntactic dependencies. German biography corpora are also semantically annotated. All but one corpus (TüBa-D/Z) are annotated automatically. The annotation pipeline applied to the German data consists of off-the-shelf tools as well as the lexicon we extracted from Wikipedia with a little effort. A few heuristics have been described which enhanced the annotation.

The fusion algorithm (Chapter 4) is tested on CoCoBi (Sec. 2.1.1). The compression algorithm (Chapter 8) is evaluated on the English compression corpus (Sec. 2.2.1) and on TüBa-D/Z (Sec. 2.1.3).

[12] The version from October 26 2008

[13] Label *cop* is excluded.

LABEL	DESCRIPTION	LABEL	DESCRIPTION
adv	adverbial modifier	*obja2*	second accusative object
app	apposition	*objc*	clausal object
attr	noun attribute	*objd*	dative object
aux	auxiliary verb	*objg*	genetive object
avz	verb prefix	*obji*	infinitive object
cj	conjunction	*objp*	prepositional object
det	determiner	*par*	“parenthesis” (intervening clause)
eth	dative subordination	*part*	subordinate particle
expl	expletive	*pn*	noun object for prepositions
gmod	genetive modifier	*pp*	prepositional modifier
grad	degree modifier	*pred*	predicate
kom	comparative	*punct*	punctuation
kon	conjuncts	*rel*	relative clause
konj	subordinate conjunction	*s*	root
neb	subordinate clause	*subj*	subject
np2	logical subject in coordination	*subjc*	clausal subject
obja	accusative object	*zeit*	temporal expression

Table 2.3: Set of dependency relations assigned by WCDG

LABEL	DESCRIPTION	LABEL	DESCRIPTION
*dep**	dependent	*amod*	adjectival modifier
aux	auxiliary	*appos*	appositional modifier
auxpass	passive auxiliary	*advcl*	adverbial clause modifier
*arg**	argument	*purpcl*	purpose clause modifier
agent	agent	*det*	determiner
comp	complement	*predet*	predeterminer
acomp	adjectival complement	*preconj*	preconjunct
attr	attribute	*infmod*	infinitival modifier
ccomp	clausal complement with internal *subj*	*partmod*	participial modifier
xcomp	clausal complement with external *subj*	*advmod*	adverbial modifier
compl	complementizer	*neg*	negation modifier
obj	object	*rcmod*	relative clause modifier
dobj	direct object	*quantmod*	quantifier modifier
iobj	indirect object	*tmod*	temporal modifier
pobj	prepositional object	*measure*	measure phrase modifier
mark	word introducing *advcl*	*nn*	noun compound modifier
rel	word introducing relative clause	*num*	numeric modifier
subj	subject	*number*	part of compound number
nsubj	nominal subject	*prep*	prepositional modifier
nsubjpass	passive nominal subject	*poss*	possession modifier
csubj	clausal subject	*possessive*	possessive *'s*
csubjpass	passive clausal subject	*prt*	phrasal verb particle
cc	coordination	*parataxis*	parataxis
conj	conjunct	*punct*	punctuation
expl	expletive	*ref*	referent
*mod**	modifier	*sdev*	semantic dependent
abbrev	abbreviation modifier	*xsubj*	controlling subject

Table 2.4: Set of dependency relations assigned by the Stanford parser

Chapter 3

Grouping Related Sentences

Sentence fusion methods operate on similar sentences. In our work, we define these as sentences sharing a significant portion of their content, although for some applications it might be reasonable to group sentences which share only an NP. For example, two sentences sharing the agent, as in (3.1) and (3.2), can be fused so that one of the input sentences becomes a relative clause (3.3):

(3.1) Paul Krugman was awarded the 2008 Nobel Prize in economics.

(3.2) Mr. Krugman is an Op-Ed columnist for The New York Times.

(3.3) Paul Krugman, who was awarded the 2008 Nobel Prize in economics, is an Op-Ed columnist for The New York Times.

Such a fusion might be particularly useful in the context of single-document summarization because in a single document sentences seldom share more than one NP and highly similar sentences are indeed very unusual. However, a simple conversion of a main clause into a relative one requires the use of words not present in the input sentences (e.g., *who* in Ex. 3.3). Apart from that, this is a relatively unchallenging transformation which is unlikely to produce an ungrammatical sentence, provided that the input is correct. In our work we want to group together or **align** sentences which are more tightly related.

Ideally, one should align sentences which concern the same event and overlap in propositions. Unfortunately, on the implementation side, this means that such an approach would require an analysis deeper than what can be achieved with existing semantic tools and methods. In this respect shallow methods such as word or bigram overlap, (weighted) cosine or Jaccard similarity are appealing as they are cheap and robust. The approach we undertake relies on such a shallow similarity measure (Section 3.2) and clusters similar sentences based on their pairwise similarity. The clustering algorithm is explained in Section 3.3.

3.1 Related Work

The task of identifying similar sentences in a pair of comparable documents is akin to the one of aligning sentences in parallel corpora which is necessary for training statistical machine translation (SMT) systems. However, sentence alignment for comparable corpora requires methods different from those used in SMT for parallel corpora. There, the alignment methods rely on the premise that a translation presents the information in the same order as in the source language. The alignment search is thus limited to a window of a few sentences. Unlike that, given two biographies of a person, one of them may follow the timeline from birth to death whereas the other may group events thematically or tell only about the scientific contribution of the person. Thus, one cannot assume that the sentence order or the content is the same in two biographies, and methods developed for parallel corpora are hardly applicable here.

Hatzivassiloglou et al. (1999, 2001) introduce SimFinder – a clustering tool for summarization systems which organizes similar text fragments into clusters. SimFinder utilizes a set of linguistic features (e.g., WordNet synsets, syntactic dependencies) and relies on annotated data to compute the similarity of two text pieces. Once pairwise similarities are computed, it uses a non-hierarchical clustering algorithm (Späth, 1985) to build clusters. In evaluation with human judges SimFinder achieved about 50% precision and 53% recall (Hatzivassiloglou et al., 1999). Since we do not have an annotated corpus at our disposal, we are looking into unsupervised methods of computing text similarity.

As a part of the DAESO[1] project, a number of shallow similarity measures were evaluated with regard to how well they can identify related sentences in a comparable corpus of Dutch news. According to the presentation available on the project website[2], weighted cosine similarity provides good results (60% of F-measure). This gives us confidence that reasonable clusters can be obtained even with a shallow unsupervised method.

Nelken & Shieber (2006) concern the task of aligning sentences in comparable documents: the gospels of the New Testament (Matthew, Mark and Luke) and the articles from the Encyclopedia Britannica. In particular, they demonstrate the efficacy of a sentence-based *tf.idf* score when applied to such very different comparable corpora. They further improve the accuracy by integrating into their algorithm sentence ordering, which is similar across the related documents in their data. For example, the encyclopedia data they use, collected and annotated by Barzilay & Elhadad (2003), consists of pairs of articles – a comprehensive and an elementary one – which all come from the Encyclopedia Britannica. Our articles come from very different sources, and the organization of biographies differs considerably even for the same person. Interestingly, their arguably simpler method outperforms the supervised SimFinder.

[1] See `http://www.daeso.nl`.

[2] `http://daeso.uvt.nl/downloads/atila08.pdf`

3.2 Similarity Measure

Before a clustering algorithm can be applied, pairwise similarities between the sentences of every pair of related documents need to be computed. We use the cosine similarity, which is widely used in information retrieval, to measure similarity (Eq. 3.4) between two sentences. In information retrieval, the weight $w_d(t)$ of a term t in the document d, belonging to the document collection D, is usually defined as its *tf.idf* (Eq. 3.5)

$$sim(d_1, d_2) = \frac{d_1 \cdot d_2}{|d_1||d_2|} = \frac{\sum_t w_{d_1}(t) w_{d_2}(t)}{\sqrt{\sum_t w_{d_1}^2(t) \sum_t w_{d_2}^2(t)}} \tag{3.4}$$

$$w_d(t) = tf_d(t) idf(t) \tag{3.5}$$

whereby the tf and idf functions are defined as follows:

$$tf_d(t) = |t|_d \tag{3.6}$$

$$idf(t) = \log\left(\frac{|D|}{|t|_D}\right) \tag{3.7}$$

Similar to Nelken & Shieber (2006), we apply these formulas to sentences. Hence in our case documents become sentences and the document collection consists of all the sentences from the related documents. The sentence similarity formula then is as follows:

$$sim(s_1, s_2) = \frac{s_1 \cdot s_2}{|s_1||s_2|} = \frac{\sum_t w_{s_1}(t) w_{s_2}(t)}{\sqrt{\sum_t w_{s_1}^2(t) \sum_t w_{s_2}^2(t)}} \tag{3.8}$$

Before applying the term weighting formula on the sentence level, we may make the following observations. On the document level, the number of times a word (or term *t*) appears in the document d, i.e., its tf_d, plays a significant role. For example, both a biography of Albert Einstein and a biography of Niels Bohr mention Niels Bohr. Naturally, the frequency of the word *Bohr* in his own biography is greater than the frequency of the same word in the biography of Albert Einstein. This can be used as a reliable indication that the former biography is a more appropriate result for the query *Niels Bohr* than the latter. However, on the sentence level it is uncommon for open-class words to appear more than one time in a sentence, because the subsequent mentions are usually pronominalized. Apart from that, there seems to be no natural correspondence between the frequency of a term in a sentence and the relevance of this sentence for the term. Consider the following example:

(3.9) Aage Bohr is the son of Niels Bohr and Margrethe Bohr.

(3.10) Aage Bohr was born in Copenhagen in 1922.

(3.11) Aage Bohr is a Danish nuclear physicist and Nobel laureate.

It would be an exaggeration to say that Sentence (3.9) is more relevant for the query *Aage Bohr* than Sentences (3.10-3.11) just because a part of the query, namely the word *Bohr*, appears three times more often there. Therefore, it makes little sense to compute the exact number of times a term (i.e., a word) appears in a sentence. Instead, an indicator function, $s(t)$, can be used, so that the weight of a term is defined as follows:

$$w_s(t) = s(t) \log \left(\frac{|S|}{|t|_S} \right) \tag{3.12}$$

where $|S|$ stands for the total number of sentences in the collection of biographies of one person, and $|t|_S$ represents the frequency of t in this collection (only one mention per sentence counts).

Furthermore, the similarity measure is enhanced with the following information:

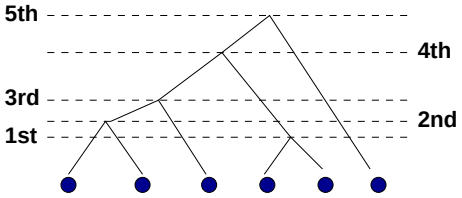
1. All references to the biographee are replaced with a special tag (*bio*).
2. We consider lemmas of words instead of their inflected forms.
3. We check with GermaNet (Lemnitzer & Kunze, 2002) whether two different words are in fact synonyms.
4. The set of terms to consider is limited to nouns, verbs (auxiliary and modal verbs excluded), adjectives and cardinal numbers (Eq. 3.13).

$$w_s(t) = \begin{cases} s(t) \log \left(\frac{|S|}{|t|_S} \right) & \text{if } pos(t) \in \{\texttt{noun}, \texttt{verb}, \texttt{adj}, \texttt{card}\} \\ 0 & \text{otherwise} \end{cases} \tag{3.13}$$

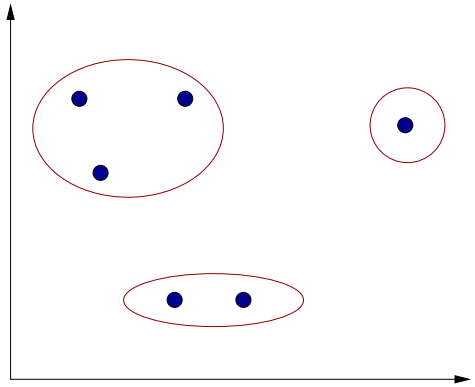
According to (3.13), a shared date is a good indicator of sentence similarity whereas the *bio* tag encountered in more than half of all the sentences is not. The pairwise sentence similarity scores are computed in this way for every pair of related biographies and are stored in a similarity matrix. Since most scores are equal to zero, the matrix is fairly sparse.

3.3 Clustering Methods

A variety of algorithms exists which cluster objects based on their pairwise similarities. One of the fundamental distinctions is related to whether the method is **hierarchical** or not. Another distinction which we will not consider in detail here concerns the decision whether an element may belong to one or more clusters. In our implementation a sentence is allowed to belong to exactly one cluster, although of course this does not have to be this way.



(a) A hierarchical bottom-up clustering of six elements



(b) Three clusters of six elements.

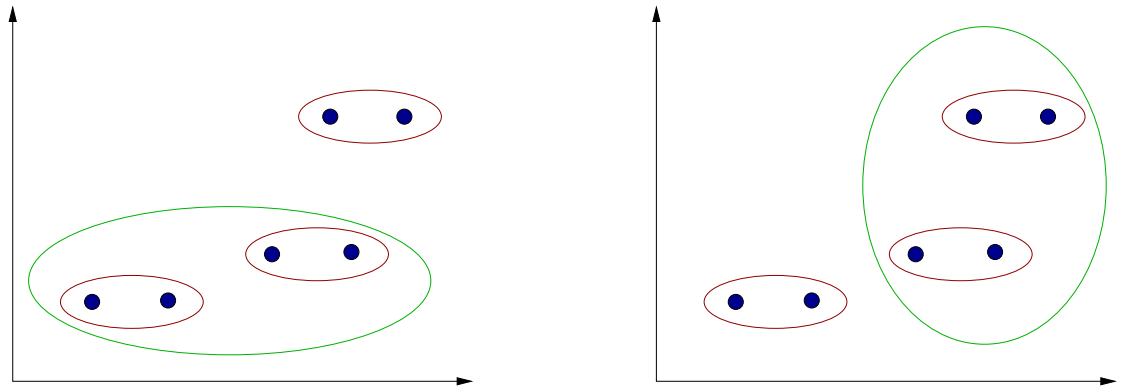
Figure 3.1: An example of six elements clustered into three groups.

3.3.1 Hierarchical Methods

Hierarchical methods proceed by iteratively building groups of elements and may start either with a single cluster including all the elements, or with a set of single-element clusters. The former approach is called **top-down**, and after every iteration it increases the number of clusters by one. The latter approach is called **bottom-up** or **agglomerative** clustering and reduces the number of clusters by merging two of them at each iteration. Top-down clustering is used less often than bottom-up clustering, which can be explained by the fact that it requires another clustering algorithm to find the best split at every iteration (Manning & Schütze, 1999).

No matter which approach is chosen, top-down or bottom-up, a tree emerges as a result of hierarchical clustering (if the clustering process has not been interrupted at some point, of course). Nodes higher in the tree represent clusters with smaller similarity. In order to get a set of clusters, one needs to break down the nodes starting from the root of the tree and moving downwards. One should stop as soon a desired number of clusters has been obtained. The tree in Figure 3.1a illustrates the point. Blue circles are objects which have been hierarchically clustered in a bottom-up manner. The dashed lines stand for the clustering levels: first the fourth and the fifth circles were grouped together, then the left two to which the third circle was later added, then the first five circles were put into one cluster, etc. Removing the root of the tree would give us two clusters: the first one including all but the rightmost circle, the second one including the last circle only. Removing the next highest node would give us three clusters {the 1st, 2nd, 3rd circles}, {the 4th and 5th circles} and {the 6th circle}.

For better visualization it is common to represent the elements to cluster as points in the space. Since in this representation proximity is a natural criterion for grouping two points in a cluster, one inverts the similarity measure so that greater similarity corresponds to a smaller distance in the space (see Figure 3.1b). At every iteration bottom-up algorithms find the two



(a) A single link clustering

(b) A complete link clustering.

Figure 3.2: Single and complete link clustering of six elements.

most similar, i.e., closest clusters and merge them. Depending on how cluster similarity is defined, one may further distinguish between the following kinds of algorithms:

Single Link Clustering. Given a pair of clusters C_1, C_2, one measures the similarity between every pair of elements c_i, c_j where $c_i \in C_1$ and $c_j \in C_2$. Then the similarity between C_1 and C_2 is defined as the maximum of all the pairs: $\max(s_i, s_j)$. A common tendency of this approach is that one gets a chain of elements in which the distance between the first and the last elements is quite large (note the green ellipse in Figure 3.2a). In the sentence space this means that given two-clause sentences *A*, *B*, *C* and *D*, such that *A* shares a clause with *B*, *B* shares a clause with *C*, and finally *C* shares a clause with *D*, and all the shared clauses are different, they would turn out to be in one group. For fusion this group would give us a possibility to generate a long five-clause sentence with no fusion done within single clauses. Thus, for us it would be better to have clusters in which elements are more tightly related so that novel clauses could be generated.

Complete Link Clustering. Here, the cluster similarity is defined in the opposite way – as the similarity of their two least similar elements. As a result, the clusters look "rounder" than the "elongated" clusters of single link clustering (see Figure 3.2b). However, complete link clustering may turn out to be too strict, so that the clusters contain very few elements. This is also undesirable because we want to have enough material from which we could later generate new sentences.

Group Average Clustering. An approach which offers a compromise between the previous two defines cluster similarity as the average of all the pairwise similarities. This way a new element joins the cluster if its average similarity to all the cluster members is high. As a result,

cases are excluded where a new element enters a large cluster because it is similar to only one of its elements. And in cases when a new element is similar to all the cluster members but one, the element is still added to the cluster.

3.3.2 Non-Hierarchical Methods

Non-hierarchical methods often start with a random partition which is further iteratively improved until no significant improvement can be achieved. Improvements can be measured, e.g., in terms of group-average similarity: a better partition is the one where the average within group similarity is higher. K-means and the Expectation Maximization algorithms are perhaps the most common examples of non-hierarchical clustering. An important question for these methods is how to determine the initial number of clusters. Sometimes there is a natural expectation of what this number should be. In cases where it is not known, a way of settling on the number of clusters is to look for k such that there is a drop in the clustering quality when one increases or decreases k.

In our experiments we have no a priori feeling for what the number of clusters should be. Moreover, we expect most sentences to stay "unclustered" (i.e., in one-element clusters). Therefore a hierarchical bottom-up approach seems to be a more appropriate way of clustering related sentences.

3.3.3 Greedy Group-Average Clustering

Our clustering algorithm is similar to group average clustering. However, in order to get larger clusters, we build them one by one so that the final clustering is not always globally optimal. We prefer to add yet another element to an existing cluster and thus make it bigger rather than to merge this element with its most similar neighbor and have two smaller clusters. For example, we prefer obtaining two clusters including three and one elements respectively to having two clusters of two elements even if the latter would have a greater group-average similarity.

Figure 3.3 presents our algorithm in a pseudo-code (with some Java flavor as the system is implemented in Java). We start by selecting the pair of most similar sentences from all the similarity tables with the GET-MOST-SIMILAR-PAIR function (see Fig. 3.4). This function returns a pair of most similar sentences provided that they come from different documents and that their similarity lies between the lower and upper bounds on similarity: $\tau < sim(s_1, s_2) < \rho$. We tuned the value of τ on a development set ($\tau = 0.1$); we set an upper bound on similarity to discard identical or nearly identical sentences ($sim(s_1, s_2) > \rho = 0.8$) and avoid generating a sentence identical to the input.

We then continue by iteratively searching for sentences which could fit in the existing

cluster best. Only sentences from documents, whose sentences are not already present in the group we are building, are considered (see the ARE-FROM-SAME-DOC function in Fig. 3.4). We add new elements provided that their average similarity to the elements of the group is above τ and is never greater or equal to ρ. The search for new clusters members terminates when no more sentences with sufficient similarity can be found. On Line 25 in Figure 3.3 the sentences of a newly created group are removed from the list of sentences from which we build groups. As a result, one sentence may belong to one cluster at most. The algorithm terminates when the similarity tables contain no more scores greater than τ.

3.4 Summary

In this chapter we presented the module for extracting similar sentences from related documents which are in our case biographies about one person. The method relies on the sentence-level *tf.idf* similarity scores. The similarity measure is enhanced with such linguistically motivated heuristics as lemmatization, coreference and synonymy, and part-of-speech information. We use a group-average algorithm to group similar sentences together. We presented a brief overview of clustering techniques and motivated our choice. The groups of related sentences are further passed to our tree generation module which we introduce in the next chapter.

function BUILD-GROUPS($sents$) **returns** list of groups

Init: $groups \leftarrow \{\}$, $g \leftarrow \{\}$
while ($g \leftarrow$ GET-MOST-SIMILAR-PAIR($sents$)) $\neq$ **null do**
  while true **do**
    $max \leftarrow \tau, best \leftarrow$ **null**, $temp \leftarrow 0$
    for all $s \in sents$ such that !ARE-FROM-SAME-DOC(s, g) **do**
      for all $t \in$g **do**
        if SIM(s, t) $< \rho$ **then**
          $temp$ += SIM(s, t)
        else
          $temp \leftarrow -1$ and **break**
        end if
      end for
      if ($temp$:= g.`size`()) $> max$ **then**
        $max \leftarrow temp$
        $best \leftarrow s$
      end if
    end for
    if $best \neq$ **null then**
      g.`add`($best$)
    else
      break
    end if
  end while
  $groups$.`add`(g)
  $sents$.`remove`(g)
end while
return $groups$

Figure 3.3: Algorithm for building groups of related sentences

function GET-MOST-SIMILAR-PAIR($sents$) **returns** list of two sentences

Init: $g \leftarrow$ **null**, $max \leftarrow \tau$
for all $s_i \in sents$ **do**
  for all $s_j \in sents$ **do**
    if s_i.getDocID() $\neq$ s_j.getDocID() **then**
      if SIM$(s_i, s_j) > max$ **and** SIM$(s_i, s_j) < \rho$ **then**
        $g \leftarrow \{s_i, s_j\}$
        $max \leftarrow$SIM(s_i, s_j)
      end if
    end if
  end for
end for
return g

function ARE-FROM-SAME-DOC($sent, group$) **returns** boolean

for all $s \in group$ **do**
  if s.getDocID() $=$ $sent$.getDocID() **then**
    return true
  end if
end for
return false

Figure 3.4: Methods used by the sentence grouping algorithm

Chapter 4

Dependency Graph Compression for Sentence Fusion

This chapter presents a novel generation method whose goal is to produce a new valid dependency tree from a group of related sentences, returned by the sentence grouping algorithm described in the previous chapter. This is a crucial part of the sentence fusion system, the other one being linearization of the generated tree (see Chapter 6). In Section 4.1 we explain the steps needed to generate a new tree and motivate each of them. Section 4.2 describes the tree transformation step, during which the dependency trees produced by the parser are transformed to facilitate tree alignment and graph compression. Section 4.3 introduces the tree alignment procedure, after which a dependency graph covering all the input trees is obtained. Finally, in Section 4.4 we describe the core part of our method which compresses the dependency graph to a dependency tree. We use integer linear programming (henceforth ILP) to obtain the highest-scoring tree. Readers unfamiliar with (I)LP may find it useful to consult Section 4.7 first before proceeding with Section 4.4.

4.1 Algorithm Overview

Given a set of similar sentences we want to produce a well-formed tree expressing relevant information from the input. One of the goals we set in the introduction is to bring fusion closer to abstraction by operating on a global representation of all the content provided. It is this global representation from which we want to extract a novel tree. In this section we explain what needs to be done to obtain such a global, semantically motivated representation.

The input to the fusion algorithm is a set of similar sentences from CoCoBi grouped together as described in the previous chapter. Recall that these sentences are annotated with part-of-speech tags and dependency relations, as well as with coreference and semantic classes.

Consider the following sentences from the biographies of André Marie Ampère[1] and their unlabeled dependency trees in Figure 4.1:

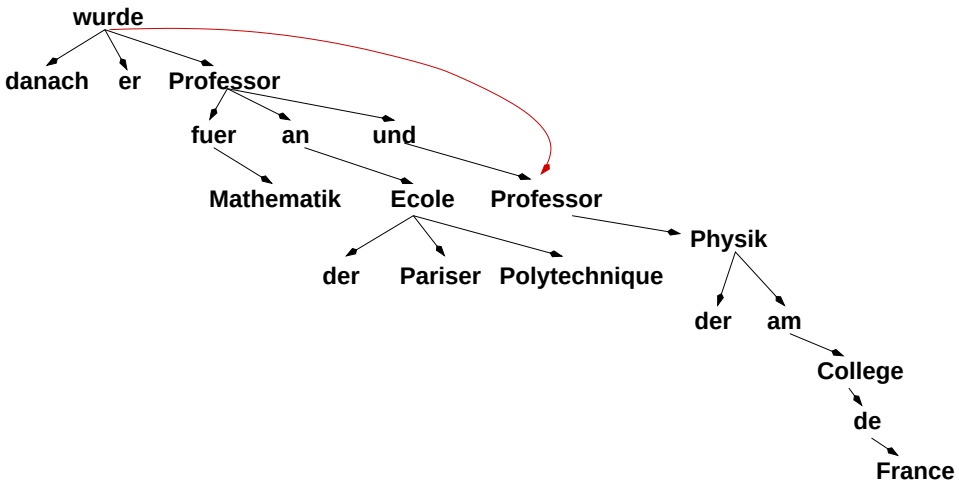
(4.1) *Danach wurde er (Ampère) Professor für Mathematik an der Pariser*
After that became he professor for mathematics at the Paris
École Polytechnique und Professor der Physik im Collège de France.
École Polytechnique and professor GEN physics in the Collège de France.
'After that he became professor of mathematics at the École Polytechnique in Paris and professor of physics at the Collège de France.'

(4.2) *Später erhielt er (Ampère) eine Professur an der École Polytechnique*
Later got he a professorship at the École Polytechnique
und am Collège de France.
and at the Collège de France.
'Later he got a professorship at the École Polytechnique and at the Collège de France.'

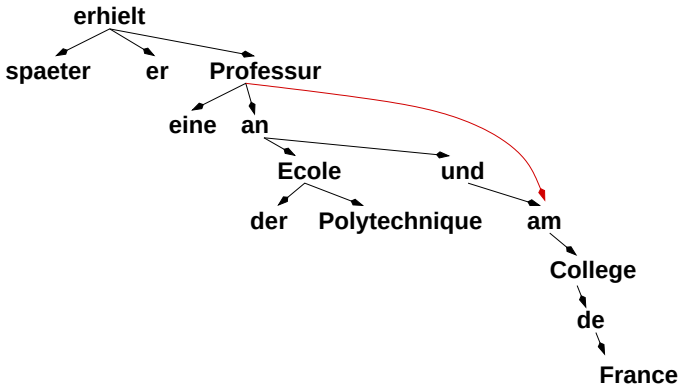
The sentences in (4.1) and (4.2) are clearly similar – both are about Ampère obtaining professorships at two famous universities in France. Moreover, the content of (4.2) is included in the content of (4.1). A part of the propositional meaning of both can be expressed as OBTAIN(AMPÈRE, PROFESSORSHIP, ÉCOLE POLYTECHNIQUE) and OBTAIN(AMPÈRE, PROFESSORSHIP, COLLÈGE DE FRANCE), where the verb *obtain* is a three-place predicate OBTAIN(WHO, WHAT, WHERE). With this representation the similarity between the two sentences is apparent. However, from the dependency representation some relations are not immediately obvious, e.g., in (4.1) it is the relation between *professor of physics* and *became* (highlighted with a red arc in Figure 4.1a). Here, *professor of physics* is embedded in a coordinated construction which is the predicate of the verb *become*. Since constituency information is not directly available in the dependency grammar, i.e., since there is no non-terminal node covering both *professor of mathematics* and *professor of physics*, the semantic relation between *professor of physics* and the verb needs to be recovered to make the similarity between the two sentences explicit. The same observation holds for (4.2) where it is the relation between the prepositional phrase *at the Collège de France* and the noun *professorship* which needs to be recovered (highlighted with a red arc in Figure 4.1b). Revealing semantic relations from a dependency representation and making semantic similarities explicit motivate the transformations we apply to the input. These transformations are described in Section 4.2.

Once the transformations are applied, the similarities between the input sentences are easier to identify, so the tree alignment part of the fusion algorithm begins. Here, the goal is to bring together all the information conveyed in the input sentences. The benefit of this alignment is three-fold:

[1] See files 10105 and 13105 in CoCoBi.



(a) The dependency structure of the sentence in (4.1)



(b) The dependency structure of the sentence in (4.2)

Figure 4.1: Dependency trees of two similar sentences about André Marie Ampère

1. We abstract from individual sentences to a global representation of **all** the input sentences.

2. Identical pieces of information are mapped together – this is important to avoid redundancy in the output.

3. Complementary information is brought together – this allows us to generate a sentence conveying information from different source sentences.

As an illustration, consider the following two sentences from the biographies of Niels Bohr[2]:

(4.3) *Nach Abitur an der Schule in Gammelholm 1903 studierte Niels Bohr*
After graduation at the school in Gammelholm 1903 studied Niels Bohr
Physik, Mathematik, Chemie, Astronomie und Philosophie an der
physics, mathematics, chemistry, astronomy and philosophy at the
Universität Kopenhagen.
University Copenhagen.
'After graduation from the school in Gammelholm in 1903 Niels Bohr studied physics, mathematics, chemistry, astronomy and philosophy at the University of Copenhagen.'

(4.4) *Er (Bohr) studierte an der Universität Kopenhagen und erlangte dort im*
Bohr studied at the University Copenhagen and obtained there in the
Jahre 1911 seine Doktorwürde.
year 1911 his PhD.
'Bohr studied at the University of Copenhagen and obtained his PhD there in 1911.'

The sentence in (4.3) contains more information about the studies of Niels Bohr, while (4.4) adds the fact that he graduated with a PhD in 1911. After alignment the identical words are mapped together and the complementary knowledge, e.g., about what Bohr studied and about his graduation with a PhD, become connected under the verb *studierte*. We describe the alignment procedure in Section 4.3.

The dependency graph includes all the information from the input sentences which might be too much to express in one single sentence. Apart from that, the structure we get after alignment is a graph and not a tree in most cases: there might be multiple incoming dependency edges originating from different input sentences. The next subtask in the fusion process is to **compress** this graph to a grammatical dependency tree. For example, given the sentences from (4.3) and (4.4), the goal is to generate a tree corresponding to the following sentence:

(4.5) *Niels Bohr studierte Physik und Mathematik an der Universität*
Niels Bohr studied physics and mathematics at the University
Kopenhagen und erlangte dort seine Doktorwürde.
Copenhagen and got there his PhD.

[2]Files 10135 and 13135 in CoCoBi.

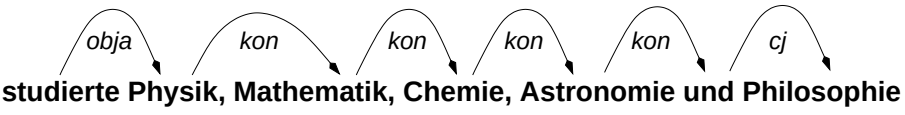


Figure 4.2: A WCDG parse of a coordinated construction

'Niels Bohr studied physics and mathematics at the University of Copenhagen and got his PhD there.'

In Section 4.4 we explain how graph compression is done.

4.2 Dependency Tree Transformation

As we have shown in the previous section, the primary motivation behind our tree transformations is to reveal semantic relations not obvious in the dependency structure. For example, this concerns coordination which is known as a weak part of dependency syntax in general (Mel'čuk, 2003) and which is treated slightly differently by different parsers. The structure assigned to a coordinated construction by the WCDG parser (see Sec. 2.1.1) is a chain where all coordinated elements starting from the second one depend on the previous one with the labels *kon* and *cj* as well as the conjunction if present (see Fig. 4.2). However, a semantic relation holds between the parent of the first word and every subsequent element in the chain (*studierte Physik, studierte Mathematik, studierte Chemie*, etc.). One of our transformations breaks such chains and attaches each of the coordinated elements to the head. This and some other transformations are semantically motivated; others facilitate further alignment, some of the transformations are necessary for the compression stage.

To illustrate the transformation process we modify the dependency tree (see Figure 4.3a) of the sentence in (4.6) by applying the transformations one by one. In our implementation we apply the transformations as soon as a suitable node has been reached while recursively traversing the tree in a depth-first manner.

(4.6) *Niels Bohr studierte Mathematik und Physik an der Universität in*
Niels Bohr studied mathematics und physics at the University in
Kopenhagen.
Copenhagen.
'Niels Bohr studied mathematics and physics at the University in Copenhagen.'

We found following transformations useful:

PREP: preposition nodes (*an, in*) are removed and placed as labels on the edges to the respective nouns (see Fig. 4.3b);

CONJ: a chain of conjuncts (*Mathematik und Physik*) is split and each node is attached to the parent node (*studierte*) provided they are not verbs (see Fig. 4.3c);

APP: a chain of words analyzed as appositions by the WCDG parser (*Niels Bohr*) is collapsed into one node (see Fig. 4.3d);

FUNC: function words such as determiners (*der*), auxiliary verbs or negative particles are removed from the tree and memorized with their lexical heads (memorizing negative particles preserves negation in the output) (see Fig. 4.3e);

ROOT: every dependency tree gets an explicit root which is connected to every finite verb with the *s* label (see Fig. 4.3f);

SEM: all occurrences of the biographee (*Niels Bohr*) are replaced with the *bio* tag (see Fig. 4.3g), alternative referring expressions are memorized with the node. All temporal expressions and named entities are collapsed into one node;

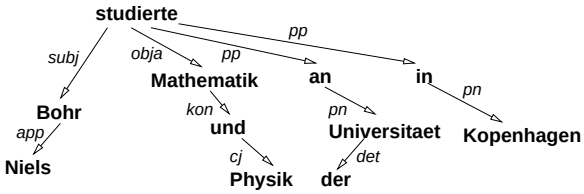
LEMMA: the open-class words (*studierte*) in the remaining nodes are replaced with their lemmas (see Fig. 4.3h). The surface form is memorized.

Note that in cases when a sentence consists of more than one clause, the resulting structure is no longer a tree because the verbs from subordinate clauses get an additional incoming edge after ROOT is applied. Usually, the transformations reduce the depth of the tree (e.g., from three to two as in Fig. 4.3) but increase the average number of siblings (e.g., from four to five as in the same Figure). The memorized auxiliary information is stored until the post-compression transformations are applied which convert transformed trees in the WCDG parser format (see Sec. 4.5).

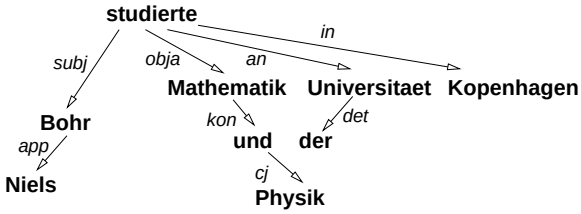
4.3 Constructing of a Dependency Graph

Once we have a group of transformed dependency trees, we aim at finding the best node alignment for those trees to further build a graph expressing all the content from the input. We use a simple, fast and transparent method and align any two nodes provided that the nodes they contain

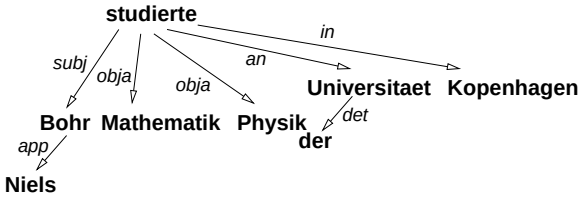
- are content words;
- have the same part-of-speech;
- have identical lemmas or are synonyms.



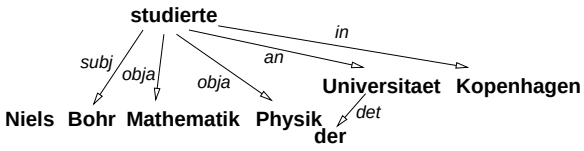
(a) Original tree



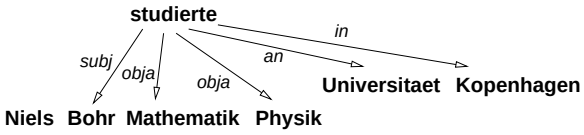
(b) After PREP



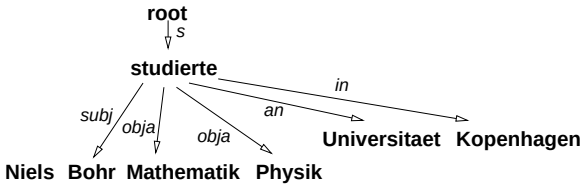
(c) After CONJ



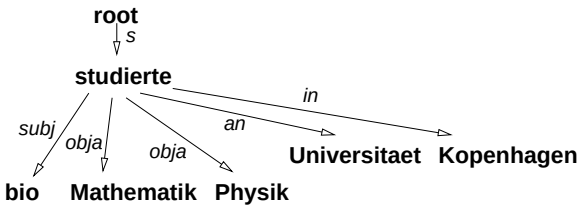
(d) After APP



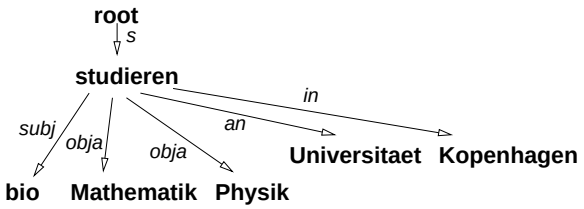
(e) After FUNC



(f) After ROOT



(g) After SEM



(h) After LEMMA

Figure 4.3: The transformations of the dependency tree of the sentence in (4.6)

We prefer this very simple method to bottom-up ones (Meyers et al., 1996; Barzilay & McKeown, 2005; Marsi & Krahmer, 2005) mainly for two reasons. Firstly, pursuing local subtree alignments, bottom-up methods may leave identical words unaligned and thus prohibit fusion of complementary information. On the other hand, they may force alignment of two unrelated words if the subtrees they root are largely aligned. Although in some cases it helps to discover paraphrases, it also considerably increases chances of generating ungrammatical output which we want to avoid at any cost. For example, even synonymous verbs such as *say* and *tell* have different subcategorization frames, and mapping one onto another would include the possibility of generating **said him* or **told to him*. In case of multiple possibilities, i.e., in cases when a word from one sentence appears more than once in a related sentence, the choice is made randomly. It should be noted, however, that such cases are extremely rare in our data.

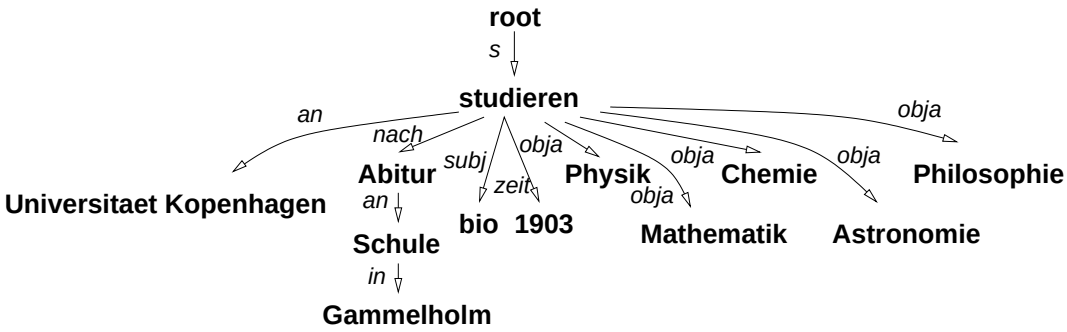
By merging all aligned nodes we get a dependency graph which consists of all the dependencies from the input trees. If the graph contains a cycle, one of the alignments from the cycle is eliminated. Root insertion during the transformation stage guarantees that the graph obtained as a result of alignment is connected. Recall the sentences (4.3) and (4.4) from Section 4.1. Figures 4.4a and 4.4b show their transformed dependency trees and Figure 4.4c presents the graph obtained as a result of their alignment (nodes shared by the input trees are in blue).

The graph we obtain covers all the dependency relations from the input sentences. Moreover, these are no longer relations between words but between entities and concepts as some nodes cover several words which may differ. For example, the node *bio* in Figure 4.4c represents an entity (namely, Niels Bohr) referred to with *er* and *Niels Bohr* in (4.3-4.4). Given that some of the tree transformations reveal implicit semantic relations, the graph is not purely syntactic but is also semantically motivated. Constructing such graphs is important because it brings sentence fusion one step closer to abstractive summarization which, as the reader might remember from the introduction, proceeds by "understanding" the text – i.e., creating a semantic representation for it – and by generating a summary from this representation.

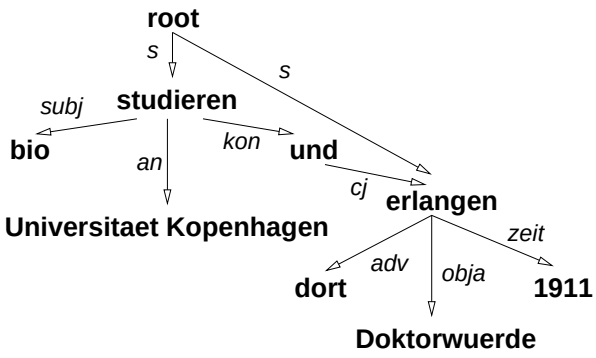
Apart from the apparent advantages, the dependency graph representation also has a number of serious disadvantages. For example, time and temporal relations are not treated properly. Consider, e.g., a set of two similar sentences such as (4.7-4.8)[3] which concern the same activity (marriage) but two different events. From the graph emerging after alignment of the respective trees (see Fig. 4.5) it is no longer possible to infer whom Albert Einstein married in which year.

(4.7) *[...] am 2. Juni 1919 heiratete er Elsa, die ihre Töchter Ilse und*
[...] on the 2nd June 1919 married he Elsa, who their daughters Ilse and
Margot mit in die Ehe brachte.
Margot VERB-PREFIX in the marriage brought.

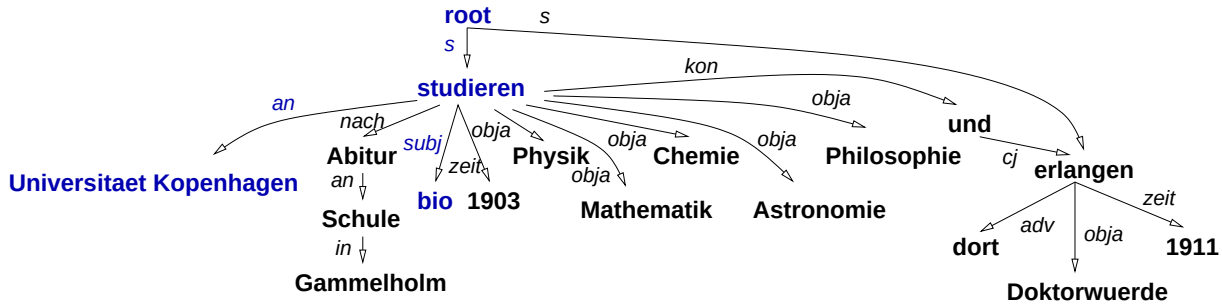
[3] See Files 10199 and 13199 in CoCoBi.



(a) The transformed tree of the sentence in (4.3)



(b) The transformed tree of the sentence in (4.4)



(c) The dependency graph obtained from the trees above

Figure 4.4: Transformed dependency trees of the sentences (4.3) and (4.4) and the graph built from them

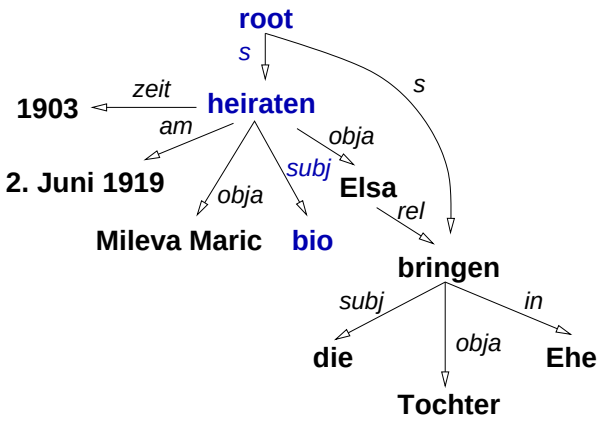


Figure 4.5: Graph built from the trees of the sentences (4.7-4.8)

'[...] on the 2nd of June 1919 he married Elsa with whom he had two daughters Ilse and Margot.'

(4.8) *1903 heiratete er Mileva Mariç, eine Mitschülerin am Polythechnicum.*
1903 married he Mileva Mariç, a classmate at the Polythechnicum.
'In 1903 he married Mileva Mariç, a classmate at Polythechnicum.'

Thus, it is important that only sentences concerning the same event are aligned and grouped together, otherwise a sentence inconsistent with the input can be generated. Fortunately, even a shallow word-based alignment algorithm turns out to be robust enough in practice, and alignment of inherently different sentences is unlikely. Still, it is important to keep in mind this limitation of our graph-based representation. Other disadvantages are due to dependency syntax in general. For example, it is impossible to express that a certain subtree modifies the proposition as a whole and not just the verb because there are no non-terminal nodes in dependency grammar.

4.4 Graph Compression

At this point we have a graph covering all the information contained in the input. Now, the next task is to get a dependency tree from this graph which we can later linearize to a German sentence. Apparently, there are many ways of getting a tree – any single edge from the graph is itself a tree, albeit a trivial one. Let us view the process of getting a new tree as a **graph compression process**. We eliminate the edges from the graph which we do not want to appear in the final tree, so that nodes which have neither incoming, nor outgoing edges also disappear. An empty tree, i.e., a tree without edges, corresponds to an empty sentence – a sentence without words. Eliminating of edges should be performed under at least the following three conditions:

Grammaticality of the output must be ensured. This condition depends to a large extent on the presence of obligatory arguments. For example, in German as well as in English every finite verb requires a subject, therefore if we decide to retain a verb we should also retain its subject. This consideration concerns not only verbs but also nouns. For example, *meeting* (*meeting **with** someone*) and *father* (*father **of** someone*) very often require a prepositional modifier to be interpreted.

Important or informative nodes should be retained in the output. The importance of a node can be computed in a generic way by favoring information appearing in more than one input sentence, or with respect to a query. For example, given a query including a certain period of time or given a *when*-question, we should retain temporal information if available (e.g., the year when a person was born) and may eliminate the location (e.g., where this person was born).

The structure of the output has to be a tree. Clearly, if we just retain a few important words with their arguments – i.e., if we fulfill the previous two conditions – the fragments might not even be connected. What we need to ensure is that the compressed structure is a tree.

In this section we present our method of simultaneously fulfilling these three and further conditions using integer linear programming.

4.4.1 Syntactic Importance Score

Given a dependency graph we want to get a new dependency tree from it. To fulfill the grammaticality condition, we want to retain obligatory dependencies (e.g., *subj*) while removing less important ones (e.g., *adv*). When deciding on pruning an argument, previous approaches either used a set of hand-crafted rules (e.g., Barzilay & McKeown (2005)), or utilized a subcategorization lexicon (e.g., Jing (2001)). The hand-crafted rules are often too general to ensure a grammatical argument structure for different verbs (e.g., *PPs can be pruned*). Subcategorization lexicons are not readily available for many languages and usually cover only verbs. For example, they do not tell us that the noun *son* is very often modified by a PP using the preposition *of*, as in *the son of Niels Bohr*, and that some NPs without a PP modifier may appear incomplete.

To overcome these problems, we decide on pruning an edge by estimating the conditional probability of its label given its head ($C(h, l)$ stands for the number of times that head node h co-occurs with label l; L represents the set of edge labels):

$$P(l|h) = P_{MLE}(l|h) = \frac{C(h, l)}{\sum_{l \in L} C(h, l)} \tag{4.9}$$

subj	obja	in (in)	an (at)	nach (after)	mit (with)	zu (to)
0.88	0.74	0.44	0.42	0.09	0.02	0.01

Table 4.1: Probabilities of *subj, obja(ccusative), in, at, after, with, to* given the verb *studieren* (*to study*)

The frequency counts are calculated from the automatically annotated German corpora described in Chapter 2, Section 2.1.

Consider the verb *studieren* (*study*) and its possible arguments. For example, $P(subj|studieren)$ – the probability that an edge outgoing from the verb *study* bears the label *subject* – is higher than $P(in|studieren)$, and therefore the subject should be retained whereas the prepositional label and thus the whole PP can be pruned, if needed. Table 4.1 presents the probabilities of several labels given that the head is *studieren* and shows that some prepositions are more important than other ones. Note that if we did not apply the PREP modification we would be unable to distinguish between different prepositions and could only calculate $P(pp|studieren)$ which would not be very informative.

In case a node contains two or more synonymous lemmas, the highest probability over all possible lemmas is selected:

$$P(l|h) = \arg\max_{h_i \in h} P(l|h_i) \tag{4.10}$$

For example, given that the verbs *wohnen* (*to live, to reside*) and *leben* (*to live, to exist*) coming from two similar sentences are aligned in the graph, the score of the *subj* edge outgoing from the node is as follows:

$$P(subj|wohnen : leben) = \max(P(subj|wohnen), P(subj|leben)) = \max(0.83, 0.52) = 0.83$$

The edges outgoing from the *rootnode* (labeled with *s*) always get a zero score. In case of unseen words or dependencies two strategies can be explored. The first one uses the part-of-speech information instead – the conditional probability of a label given the PoS of the head is calculated (the modified formula no longer defines a probability distribution, hence we replace probability with *Score*):

$$Score(l|h) = \begin{cases} P(l|h) & \text{if } P(l,h) > 0 \\ P(l|pos(h)) & \text{otherwise} \end{cases} \tag{4.11}$$

However, we prefer not to fall back on PoS information but to update the scores. Updating the frequency database with the counts from a new document does not require much effort because one pass over the document is needed anyway to calculate the word informativeness score introduced next.

4.4.2 Word Informativeness Score

According to the second condition listed above, we want to retain informative words in the output tree. There are many ways in which word importance can be defined. Here, we use a formula introduced by Hori & Furui (2004) to define the word's significance score:

$$I(w) = f_w \log \frac{F_A}{F_w} \tag{4.12}$$

w is the topic word (either noun or verb), f_w is the frequency of w in the aligned biographies, F_w is the frequency of w in the corpus, and F_A is the sum of frequencies of all topic words in the corpus. Personal pronouns get a high default score. Just like with the syntactic importance score, in case of different lemmas merged in one node, the word informativeness score is defined as the maximum from all the scores:

$$I(w) = \arg\max_{w_i \in w}(I(w_i)) \tag{4.13}$$

Note that in the context of topic-oriented summarization word importance could be defined differently, e.g., as relatedness of a word to the topic.

4.4.3 Generating a Tree from the Graph

4.4.3.1 Tree Extraction as an Optimization Problem

Now, we formulate the task of getting a tree from a dependency graph as an optimization problem and solve it with ILP[4]. In order to decide which edges of the graph to remove, for each directed dependency edge from head h to word w we introduce a binary variable $x^l_{h,w}$, where *l* stands for the label of the edge:

$$x^l_{h,w} = \begin{cases} 1 & \text{if the dependency is retained} \\ 0 & \text{otherwise} \end{cases} \tag{4.14}$$

The goal is to find a subtree of the graph which gets the highest score of the objective function (4.15) to which both the syntactic importance score ($P(l|h)$) and the word informativeness score ($I(w)$) contribute:

$$f(X) = \sum_{x \in X} x^l_{h,w} P(l|h) I(w) \tag{4.15}$$

The objective function is subject to four types of constraints presented below.

[4]We use `lp_solve` in our implementation `http://sourceforge.net/projects/lpsolve`. For a basic introduction to linear programming see Section 4.7.

4.4.3.2 Structural Constraints

STRUCTURAL constraints allow us to get a tree from the graph and thus fulfill the final condition from Section 4.4: (4.16) ensures that each word has one head at most. (4.17) ensures connectivity in the tree. W stands for the set of graph nodes minus the root, i.e. the set of words. The resulting tree is always rooted in the *rootnode* inserted after the ROOT transformation (see Sec. 4.2).

$$\forall w \in W, \sum_{h,l} x^l_{h,w} \leq 1 \tag{4.16}$$

$$\forall w \in W, \sum_{h,l} x^l_{h,w} - \frac{1}{|W|} \sum_{u,l} x^l_{w,u} \geq 0 \tag{4.17}$$

Recall the only requirement for the graph built after node alignment (Section 4.3): the graph must not contain any cycles. This is a very important requirement because it guarantees that the constraints (4.16-4.17) above suffice to get a tree. Otherwise an unconnected graph containing a cycle might be output.

Constraint (4.18) restricts the size of the resulting tree to α words.

$$\sum_{x \in X} x^l_{h,w} \leq \alpha \tag{4.18}$$

The value of α ($\alpha = \min(0.\bar{6}|W|, 10)$) is determined empirically so that the generated tree is about the same size as the input trees. In cases when the dependency graph is large, not more than ten dependencies are permitted. Note that this does not imply that the generated sentence would contain only ten words as function words are excluded from the graph. Apart from that, appositions are collapsed into one node; e.g., *bio* may later become *Niels Henrik David Bohr*. Without (4.18) the algorithm would probably find the maximum spanning tree connecting all the nodes in the graph.

4.4.3.3 Syntactic Constraints

SYNTACTIC constraints ensure the syntactic validity of the output tree and explicitly state which arguments should be retained. We have only one syntactic constraint which guarantees that a subordinating conjunction (*sc*) is retained (4.19) if and only if the clause it belongs to serves as a subordinate clause (*sub*) in the output.

$$\forall x^{sc}_{w,u}, \sum_{h,l} x^{sub}_{h,w} - x^{sc}_{w,u} = 0 \tag{4.19}$$

For example, given (4.20) as one of the input sentences[5], we do not want to compress the graph to the tree corresponding to (4.21).

(4.20) *1811* *veröffentlichte* *er (L. R. A. C. Avogadro)* *seine* *Hypothese,* *dass* *gleiche*
In 1811 published he his hypothesis, that equal
Volumina *von* *Gasen* *unter* *gleichen* *Bedingungen* *die* *gleiche* *Anzahl* *von*
volumes of gases under equal conditions the same amount of
Molekülen *enthalten.*
molecules contain.
'In 1811 he published his hypothesis that equal volumes of gases contain the same amount of molecules under equal conditions.'

(4.21) **Dass* *gleiche* *Volumina* *von* *Gasen* *unter* *gleichen* *Bedingungen* *die* *gleiche*
That equal volumes of gases under equal conditions the same
Anzahl *von* *Molekülen* *enthalten.*
amount of molecules contain.
'That equal volumes of gases contain the same amount of molecules under equal conditions.'

We want to avoid syntactic constraints wherever possible to make the method easily portable to other languages – this is one of the goals articulated in the introduction. For that reason, we do not introduce any hard constraints for verb arguments as the number of rules can become exceedingly large. For example, every finite verb requires a subject, the verbs *go*, *depart* require an argument with a locational meaning, the verb *ask* requires a NP, a VP or a clause as its object, etc. In our implementation the task of keeping the right arguments is shifted to the objective function completely. There, it is the syntactic score which is responsible for promoting obligatory arguments. However, it is perfectly possible to complement the syntactic score with syntactic constraints and thus amend grammaticality if the algorithm consistently treats certain constructions poorly. At the moment, we do not undertake such a complementary approach in order to see how well the fusion method performs when grammaticality is ensured **solely** with the syntactic importance score and the objective function.

4.4.3.4 Semantic Constraints

SEMANTIC constraints restrict coordination to semantically compatible elements. The idea behind these constraints can be illustrated with the following two related sentences (see Fig. 4.6):

(4.22) He studied math.

(4.23) He studied physics.

[5] See file 10113 in CoCoBi.

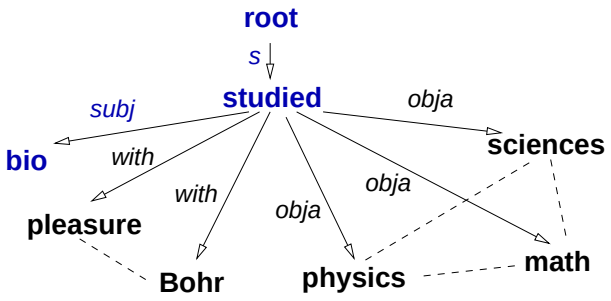


Figure 4.6: Graph obtained from the sentences *He studied sciences with pleasure* and *He studied math and physics with Bohr*

So, the output may unite the two words under coordination:

(4.24) He studied math and physics.

The situation is different with the following very similar sentences:

(4.25) He studied physics.

(4.26) He studied sciences.

Here, the arguments should not be united, because *sciences* is the generalization of *physics*, and the union of nouns which are in ISA relation should be prohibited in order to avoid generation of the anomalous sentence in (4.27).

(4.27) #He studied physics and sciences.

Now consider the following two examples:

(4.28) He studied with pleasure.

(4.29) He studied with Niels Bohr.

The union of the two noun phrases, *pleasure* and *Niels Bohr*, under the same preposition would produce an anomalous sentence with syllepsis because the two are semantically totally unrelated (4.30):

(4.30) #He studied with pleasure and Bohr.

A proper treatment of such examples is important because, given that there are several possibilities for the direct object and that the respective edges get high scores, the objective function would try to retain as many direct objects as possible. A straightforward solution could be to

require that at most one argument of every kind should be included. This would solve the problem but also make the method too restrictive. Therefore, we prefer to allow argument combination provided that semantic constraints are not violated. In particular, to formalize the intuitions illustrated with the examples (4.25-4.27) and (4.28-4.30) we introduce additional variables $y^l_{w,u}$ (represented by dashed lines in Fig. 4.6):

$$y^l_{w,u} = \begin{cases} 1 & \text{if } \exists h, l : x^l_{h,w} = 1 \wedge x^l_{h,u} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{4.31}$$

We also define two functions *hm(w,u)* and *rel(w,u)*.

hm For two edges sharing a head and having identical labels to be retained we check in GermaNet (Lemnitzer & Kunze, 2002) and in the taxonomy derived from Wikipedia (Kassner et al., 2008) that the words they are pointing to are not in the *h*yponymy or *m*eronymy relation (4.32). *hm(w,u)* is a binary function which returns *1* if one of the relations holds and *0* otherwise:

$$\forall y^l_{w,u}, hm(w,u) y^l_{w,u} = 0 \tag{4.32}$$

rel If the dependents are nouns, we also check that their semantic relatedness as measured with WikiRelate! (Strube & Ponzetto, 2006) is above a certain threshold (4.33). We empirically determined the value of $\beta = 0.36$ by calculating an average similarity of coordinated nouns in the corpus. *rel(w,u)* returns a value from $[0, 1]$:

$$\forall y^l_{w,u}, (rel(w,u) - \beta) y^l_{w,u} \geq 0 \tag{4.33}$$

Constraint (4.32) prohibits that *physics* (or *math*) and *sciences* appear together since, according to GermaNet, *physics* (*Physik*) is a hyponym of *science* (*Wissenschaft*). Constraint (4.33) prevents taking both *pleasure* (*Freude*) and *Bohr* because *rel(Freude,Bohr)* $= 0.17$. Since *math* and *physics* are neither in ISA, nor part-of relation and are sufficiently related (*rel(Mathematik, Physik)* $= 0.67$) they can become conjuncts.

Verb coordination is generally prohibited unless it is found in one of the input sentences. This is done because, unlike noun coordination, verb coordination often implies temporal and/or discourse relations such as precedence or consequence.

4.4.3.5 Meta Constraints

META constraints (equations (4.34) and (4.35)) guarantee that $y^l_{w,u} = x^l_{h,w} \times x^l_{h,u}$ i.e. they ensure that the semantic constraints are applied only if both the labels from h to w and from h

sim^{all}	sim^{def}	sim^{NZ}
0.33	0.36	0.54

Table 4.2: Average similarity of coarguments calculated from 200 WikiBiography files

to u are retained.

$$\forall y^l_{w,u}, x^l_{h,w} + x^l_{h,u} \geq 2y^l_{w,u} \tag{4.34}$$

$$\forall y^l_{w,u}, 1 - x^l_{h,w} + 1 - x^l_{h,u} \geq 1 - y^l_{w,u} \tag{4.35}$$

A note on the similarity threshold β. Table 4.2 presents the similarity values calculated on 200 files from WikiBiography. For every two coordinated nouns their similarity was calculated with the Wu & Palmer measure (Wu & Palmer, 1994) which has been shown to work particularly well for German (Ponzetto & Strube, 2007b). From 908 pairs the similarity could not be computed in 69 cases, i.e., one of the words in the pair did not have an article on Wikipedia. From the rest of 839 pairs, in 282 cases the similarity was zero (33.6%) which might be due to page disambiguation errors. On average, similarity greater than zero was as high as 0.54. Average similarity, when defined (i.e., zero or greater) is 0.36. Calculated on all the pairs, the average similarity is 0.33. We thus use the latter value as the similarity threshold β.

4.5 Post-Compression Transformations

The tree which emerges as a result of graph compression does not explicitly contain all the grammatical information. Recall the transformations described in Section 4.2 which transform the output of the WCDG parser into a semantically motivated format. The goal of the transformations we apply **after** compression is to revert those done **before** alignment and to bring the tree into the WCDG format.

PREP^{-1}: preposition labels are removed and an explicit preposition node is inserted with an incoming edge labeled *pp* and an outgoing edge labeled *pn*;

CONJ^{-1}: nodes sharing a head and labels on their incoming edges are put into a chain with the node *und* (*and*) between the last two. The order of the nodes is random;

FUNC^{-1}: function words such as determiners and negation memorized with their heads are inserted in the tree with their respective labels (*det*, *neg*);

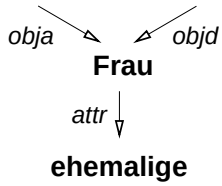


Figure 4.7: A fragment of a graph covering *seine ehemalige Frau* and *seiner Frau*

ROOT^{-1}: the root of the tree is removed;

After all the transformations are applied, the tree has a "normal" WCDG structure and can be passed to the linearizer (see Chapter 6). All of the nodes store their possible surface realizations (e.g., pronominal or full names, synonyms).

4.6 Possible Extensions

Portability to Other Languages. In the section introducing the syntactic constraints (Sec. 4.4.3.3) we explained why we try not to add syntactic constraints wherever possible. Apart from the fact that writing syntactic rules requires a lot of human labor, this would hinder the portability of the method to other languages. Indeed, the hard constraint on the syntactic structure of English clauses is that there should always be an overt subject. This constraint does not hold, e.g., in Slavic languages. Unlike hard constraints, the syntactic importance score helps to distinguish more important arguments from less important ones and replaces the binary scheme "obligatory vs. optional arguments" with a scale where all the values from zero to one are possible. Thus, the main prerequisite for the method to be applicable to a certain language is that there are a dependency parser and a corpus available.

One issue which has received little attention in the present work and which is needed for proper fusion in some languages is grammatical agreement. We will illustrate the point with a German example. Given two sentences containing the NPs *seine ehemalige Frau* (*his former wife*, accusative case) and *seiner Frau* (*his wife*, dative case), and given a graph covering these two sentences, this graph contains the fragment presented in Figure 4.7. Now if the edges with the *objd* and the *attr* labels are retained and the edge labeled *obja* is removed, a problem arises: in dative case the adjective *ehemalige* should take the form *ehemaligen*. However, such a surface form is not present in any of the input sentences and therefore cannot be included in the input. For languages like English or German where there is a small amount of agreement the problem is not severe. For languages with richer morphology it would be necessary to add a module responsible for appropriate word realization, e.g., number, gender and case declination.

The semantic constraints we introduced in Section 4.4.3.4, albeit universal by nature, do require external resources. Unfortunately, these resources are available for very few languages only. Moreover, although WordNets are being developed for many languages, currently the coverage of most of them is significantly smaller than for English. In the first place this is due to the high costs of human labor associated with the development of such a lexical database. On the contrary, Wikipedia has demonstrated exponential growth and is at the moment available in more than 250 languages. This gives us a hope to expect that for many languages in a few years there will be automatically extracted taxonomies and there will be means of calculating semantic relatedness automatically (Ponzetto & Strube, 2007a; Nastase & Strube, 2008; Ponzetto & Navigli, 2009).

Topic Oriented Sentence Fusion. The formula (4.12) from Section 4.4.2 is designed for generic sentence fusion: it favors words which are important for a certain document collection as well as for a certain sentence. However, in many applications it is more desirable to generate a new sentence with respect to a certain topic. In such cases it would be more adequate to compute the word importance by taking the words from the possibly extended topic into account. Since the word importance score is one of the (optional) components of the system and is not the base of the method, it could be defined differently depending on the user needs. One possible way to to do this would be to define the word informativeness as the semantic relatedness of the word to the topic as computed with WikiRelate! or other software for measuring semantic relatedness.

Adding Discourse Constraints. One of the advantages of the ILP formulation is that adding further constraints requires relatively little effort. Currently, our system supports generation of single sentences. Of course, independent generation of sentences for a summary is unrealistic given that it is the structure of discourse which influences, e.g., the choice of referring expressions or the use of discourse markers. Without taking into account what has been generated already, the system cannot guarantee that two fused sentences do not repeat the same information. Although repetitions are unlikely given that we group similar sentences and do not include a sentence in more than one group, still, it would be useful to have a way of stating whether certain information has appeared earlier in the summary. For example, we may want to prohibit the simultaneous retaining of the verb and two of its arguments given that this construction has appeared in an earlier sentence. Similarly, to improve summary coherence we might want to indicate which words are more appropriate given the immediately preceding context.

Generating Unseen Constructions. Sentence fusion is in essence an extractive method. It is definitely more powerful than sentence extraction because it can generate sentences which

are not present in the input. However, since sentence fusion operates on the dependency structures of given sentences, fused sentences are composed solely of extracted dependencies. For sentence fusion to be closer to abstraction, it should be able to generate constructions not present in the input. Hence, a possible extension of deFuser could be augmenting the dependency graph obtained after tree alignment with dependencies found in a corpus (see Wan et al. (2009) for a similar idea). One of the foreseeable difficulties is to add dependencies which would not distort the meaning intended in the input. For example, it would be harmful to add the *subj* edge from the verb to the accusative object and the *obja* edge pointing to the subject without passivizing the verb.

4.7 Integer Linear Programming in NLP

In this section we explain the gist of ILP and then give a short overview of NLP applications which utilize ILP.

4.7.1 (Integer) Linear Programming

Linear programming, or LP, is an optimization technique which allows to find an optimal solution for a set of variables under given constraints where the relationships between those variables are expressed with linear (in)equalities only (Cormen et al., 2001, Chapter 29). The three founders of LP are Leonid V. Kantorovich, who developed linear programming problems in 1939, George B. Dantzig, who invented the simplex method of solving LP problems in 1947, and John von Neumann, who also developed the theory in the same year. Kantorovich developed the LP approach when working as a consultant at a plywood factory at the age of 26, and at that time already recognized its potential for a vast number of problems in economics. Later, in 1975, he won the Nobel Prize in economics for his contribution to the study of LP.

An LP problem in the standard form is stated as follows: given an objective function f defined over a set of non-negative variables $X = \{x_1, x_2, ...x_n\}$, such that

$$f(X) = \sum_{i=1}^{n} c_i x_i = c_1 x_1 + c_2 x_2 + ... + c_n x_n$$

and a set of linear constraints of the form:

$$\sum_{i=1}^{n} a_i x_i \leq b$$

LP finds the solution which maximizes $f(X)$. As an illustration, one may think of an amount of money (say, 10,000 euros) which can be invested in n different ways, such that each of the investments promises the profit of c_i in a year. That is, investing x_i in the *i*-th way guarantees

you $c_i x_i$ euros in a year. In this context the goal is to find a split of your money which maximizes the overall profit. Without any constraints the solution is obvious: one should select the possibility with the highest profit and invest all money there. However, given that there are constraints on your decisions, the solution might become less obvious. In the money-investing scenario possible constraints are (i) the minimal amount one has to put into charity, (ii) the maximum limit on all the risky investments together, or (iii) that some investments lie in a range provided by a company that accepts such investments. Constraints which involve the product of two variables are forbidden.

The simplex method developed by G. B. Dantzig allows to find the best solution to an LP problem by moving over the vertices of the convex polytope defined by the constraints in the *n*-dimensional space of variables. The best solution, if it exists, can always be found on a vertex; also it is possible to move along the edges of the polytope such that the objective function increases with every vertex until a local maximum is reached. Once it is found, i.e., the vertex is found all of whose neighbors yield worse results, the globally optimal solution is obtained. Although the simplex method has poor worst-case behavior, it is reported to be efficient in practice. Alternative methods include the ellipsoid and the interior point methods.

Although a huge number of problems can be formulated with LP, there is also a large number of situations where the variables are required to be integers. For example, one may have to decide how to assign *N* people to *n* tasks to maximize the overall productivity under a set of constraints. Possible constraints could be (i) at most one task can be left with no one working on it, (ii) at most *k* people can work on a certain task because there are only *k* computers with the right software installed, or (iii) task two depends on the results of task one, and thus there should be fewer people working on task two. Interestingly, one cannot simply solve an analogous LP problem and round the solutions. For example, given two variables such that $x_1 + x_2 = 3$ and given that the optimal solution is $x_1 = x_2 = 1.5$, it is unclear what the integer solution to this problem should be – $\{x_1 = 1, x_2 = 2\}$, $\{x_1 = 2, x_2 = 1\}$, or something else. Hence, alternative methods have been developed to solve ILP problems, such as the branch and bound or cutting-plane methods which share the idea of relaxing the integral constraints (Roth & Yih, 2004). It has been shown that solving integer linear problems is NP-hard but that there are important subclasses of ILP problems where the optimal solution can be found efficiently.

Yet another important subclass of ILP is called binary integer programming (BIP) where the requirement is that every variable is from the set $\{0, 1\}$. Such problems emerge naturally when one needs to determine which decisions from a set of possibilities to make. For example, given a wardrobe full of clothes, one can formulate a BIP problem answering the question *"What should I put on today so that I look best?"*. The assumption here is that the person has a precise idea about how well every piece of clothes suits him/her expressible in real numbers. For example, a vintage hat and a little black dress get the score of 100 each, whereas an old

T-shirt with a tomato-sauce stain gets a negative score. The possible constraints could be that (i) one cannot wear more than two socks at the same time, (ii) one cannot go out naked, (iii) the vintage hat and the little black dress are mutually exclusive, or (iv) the total weight of the clothes one wears should not exceed ten kilos. Recall that in (I)LP only linear (in)equalities are permitted. However, in BIP one can find a way around this condition. For example, $x_1 \times x_2 = x_3$ can be reformulated with two inequalities: $x_1 + x_2 \geq x_3$ and $1 - x_1 + 1 - x_2 \geq 1 - x_3$. Unfortunately, BIP is also known to be NP-hard.

4.7.2 Use of ILP in NLP

Dras (1997) was perhaps the first one to utilize ILP for an NLP task – paraphrase generation. Since then ILP has been applied to discourse ordering (Althaus et al., 2004), semantic role labeling (Punyakanok et al., 2004), simultaneous entity and relation classification (Roth & Yih, 2004), natural language generation (Marciniak & Strube, 2005; Barzilay & Lapata, 2006) and parsing (Riedel & Clarke, 2006), sentence compression (Clarke & Lapata, 2006b) and anaphora resolution (Denis & Baldridge, 2007; Klenner, 2007). There is a growing interest in using ILP which is reflected in the organization of a workshop devoted to the use of ILP in NLP[6]

The popularity of ILP in NLP can be attributed to several factors. Firstly, it helps overcome an important disadvantage of the pipeline approach which has been recognized in many NLP applications: certain mistakes and apparent contradictions could be avoided if earlier modules could get feedback from later modules (Roth & Yih, 2004; Marciniak & Strube, 2005). Secondly, it helps to integrate global constraints which are otherwise neglected: e.g., transitivity in coreference sets (Klenner, 2007), or global constraints on syntactic structure (Clarke & Lapata, 2006a). The main concern when using ILP is whether the optimal solution can be found efficiently. For example, if the number of constraints grows exponentially with the size of the problem (i.e., the number of variables), it is a good indication that ILP is not the best choice for the task.

4.8 Summary

In this chapter we presented the core part of our fusion method which takes a set of parsed related sentences as input and produces a novel dependency tree. This tree is generated by compression from a dependency graph which is built from the transformed dependency trees of the source sentences. We formulated the tree generation task as an optimization problem and solved it with integer linear programming (ILP). We showed how grammaticality of the

[6] See the workshop's website `http://www-tsujii.is.s.u-tokyo.ac.jp/ilpnlp/`.

syntactic structure can be enforced with a few rules and statistics obtained from a parsed corpus. This makes our approach different from the methods which rely heavily on human-crafted resources (e.g., a subcategorization lexicon) or require a complex set of grammar rules to ensure grammaticality. To avoid at least some of semantic anomalies, we also checked semantic compatibility of co-arguments which has not been previously done. For this we utilized GermaNet and knowledge automatically extracted from Wikipedia. The approach can be ported to other languages provided there is a sufficiently large corpus and a reliable dependency parser. It would also benefit from a large taxonomy of concepts. For some languages a morphological component is required to handle grammatical agreement. The presented approach is designed for generic sentence fusion although it can be adapted to topic-oriented fusion. The generic formula for word informativeness should then be replaced with a topic-oriented one. Another property of our fusion method is that it can be applied to sentence compression. In Chapter 8 we explain how this can be done. In Chapter 6 we demonstrate how a newly generated dependency tree can be converted to a grammatical sentence.

Chapter 5

Filling the Sentence Initial Position

This chapter lays the basis for the tree linearization algorithm (discussed in the next chapter) and presents a study whose modest goal is to investigate the properties of the sentence initial position in German. Our thrust here is to argue that the sentence initial position has a special status and that the choice of what to place there influences the perceived fluency of text. As a consequence, the findings of our study are of relevance to systems designed to generate appropriate word order in German and have a direct impact on the design of our tree linearization method. In the following we are going to talk about **text coherence** which we are going to define less technically as text readability. While **global** coherence can be attributed to the reader (Graesser et al., 1994), **local** coherence is a property of the text itself (McNamara et al., 1996). Among the many factors which account for local coherence we focus here on what should be placed in the beginning of the sentence to make the transition between adjacent sentences as smooth as possible.

Starting with linguistic preliminaries (Sec. 5.1), we come to our hypothesis (Sec. 5.2) which we formulate in terms of information structure. We further present evidence in support of our hypothesis revealed in a corpus study (Sec. 5.3) as well as in an experiment with native speakers (Sec. 5.4). Moreover, we show that even texts produced by human writers can be improved. Finally, we demonstrate how to automatically select the best candidate for the sentence initial position (Sec. 5.5) and give an overview of related work (Sec. 5.6).

5.1 Theoretical Preliminaries

There exists a large body of work in theoretical linguistics about the syntactic organization and information structure of German clauses. We briefly outline particularities of German clause organization and introduce relevant work on information structure and discourse status.

5.1.1 Topological Fields

The division of the German clause into topological fields (Drach, 1937) can be explained by means of the verbal bracket (*Satzklammer*). The finite verb (or complementizer) constitutes the left verbal bracket. The right verbal bracket is formed by any other part of the verb, i.e., a separable prefix, past participle or the infinitive dependent on a modal verb. If there is no second part of the verb the right verbal bracket coincides with the end of the clause. Usually only one constituent occupies the position left of the left verbal bracket. This position is called *Vorfeld (VF)*. Everything between left and right verbal bracket constitutes the *Mittelfeld (MF)*. If there is anything between the right verbal bracket and the end of the clause, it constitutes the *Nachfeld (NF)*. The VF/MF division is normally found in main clauses; in hypotactic clauses there is no VF. In all the examples in this chapter the VF and the MF are enclosed by square brackets. In (5.1) *Marie Curie* is in the VF, *wurde* the left verbal bracket, and *geboren* the right verbal bracket. Everything between *wurde* and *geboren* is in the MF.

(5.1) *[Marie Curie] wurde [am 7. November 1867 in Warschau] geboren.*
Marie Curie was on the 7th November 1867 in Warsaw born.
'Marie Curie was born in Warsaw on the 7th of November 1867.'

5.1.2 Information Structure

Due to divergences in terminology, information structure is notoriously difficult to talk about (see Levinson (1983), p. x). It is well-known that there is no uniform definition of its central concept – the **topic**. Jacobs (2001) explains this fact by the multi-functional nature of topic-comment constructions and identifies their four basic attributes. In Jacobs' view, the diversity of definitions originates from different opinions on which attribute should be the core one. Different theories agree on prototypical cases, but may disagree when a constituent exhibits only one of the attributes of the prototype. Given a syntactic phrase (X,Y) with the immediate constituents X and Y, Jacobs distinguishes four attributes:

Information separation: X is informationally separated from Y, iff semantic processing of utterances (X,Y) involves two steps, one for X and one for Y (see Hockett (1958));

Predication: X is the semantic subject of the semantic predicate Y, iff X specifies a variable in the semantic valency of Y, and there is no such Z that (i) Z specifies a variable in the semantic valency of an element in Y and (ii) Z is hierarchically higher in semantic form than X;

Addressation: X is the address for Y, iff X marks the point in the speaker-hearer knowledge where the information carried by Y has to be stored at the moment of the utterance;

Frame-setting: X is the frame for Y, iff X specifies a domain of (possible) reality to which the proposition expressed by Y is restricted.

Information separation and predication hold for almost all cases we will discuss and thus do not play a crucial role here. Therefore, we will primarily concentrate on the addressation and frame-setting dimensions. The former is similar to Reinhart's (1981) topic as aboutness view which can be traced back to Strawson (1964). The latter, i.e., the frame-setting dimension, originates from Chafe (1976) who, similar to Jacobs, points out that the functions of topics are diverse. Chafe also defines **real topics**, which

> "are not so much 'what the sentence is about' as 'the frame within which the sentence holds'." (Chafe, 1976, p. 51)

The sentences (5.2-5.4), taken from Jacobs (2001), illustrate what is meant by the **addressation (TA)** and the **frame-setting (TF) topics** respectively:

(5.2) Let me tell you something about Peter.

(5.3) Yesterday he went to the police and told them everything.

(5.4) In my dream, Peter was a crocodile.

Because of the preceding sentence, (5.3) is clearly about *Peter*, who is the TA of the sentence. In (5.4), *in my dream* is the domain where the proposition *Peter was a crocodile* holds and is, therefore, the TF of the sentence.

We are interested in where the TA and the TF should be located in a German sentence so that it can be processed as smoothly as possible. Sentence initial position, being cognitively prominent (Gernsbacher & Hargreaves, 1988), has been recognized as a good place for the topic across different languages (Vallduví & Engdahl, 1996) and in German in particular (Molnár, 1991). Unlike this widespread opinion, Frey (2004), who defines the topic in terms of aboutness, i.e., in the TA sense, has demonstrated that there is a designated position for TAs, namely

> "[...] in the middle field of the German clause, directly above the base position of sentential adverbials". (Frey, 2004, p. 15)

The latter are adverbials which express the speaker's assessment of an eventuality (e.g., *fortunately, apparently, certainly*). The following sentence (5.5) illustrates this point:

(5.5) *[Im Jahr 1891] konnte [sie* glücklicherweise *mit dem Studium] anfangen.*
In the year 1891 could she *fortunately* with the studies begin.
'Fortunately, she could begin her studies in 1891.'

In this example there is a temporal adjunct, which has a frame-setting function, in the VF. The addressation topic referring to the person is in the beginning of the MF followed by a sentential adverbial (in italics). Since nothing but the TA can be placed before the sentential adverbial, Frey (2004) claims this place to be the TA's designated position. Insofar, the rule neither prohibits the topic to appear in the VF, nor says anything about cases without a sentential adverbial. It does not state which order leads to a more fluent sentence. So, another variant of the sentence in (5.5) as well as two very similar sentences are perfectly acceptable (5.6-5.8):

(5.6) [Sie] konnte [*glücklicherweise* mit dem Studium im Jahr 1891] anfangen.

(5.7) [Sie] konnte [mit dem Studium im Jahr 1891] anfangen.

(5.8) [Im Jahr 1891] konnte [sie mit dem Studium] anfangen.

Thus, since the latter two variants lack a sentential adverbial, the rule does not predict which of them should be preferred over the other one. As neither Jacobs nor Frey place emphasis on the preferred order of the two kinds of topics, it is of interest to check whether there is any preference at all. If so, the conditions under which these preferences hold should be identified.

5.1.3 Discourse Status

There exists another view on the topic as a measure of the entity's salience. An example of this view is Givón (1983) whose topic is very similar to the notion of the backward-looking center in the Centering model (Grosz et al., 1995). Since we define the topic differently, we exclude the discourse status of the referent from the definition of the topic. Instead, we distinguish between what has been introduced by Chafe (1987) and later adopted by Lambrecht (1994) as **active**, **semi-active/accessible**[1] and **inactive** referents:

Active referents are in a person's focus of consciousness;

Accessible referents are in a person's peripheral consciousness;

Inactive referents are in the long term memory, neither focally nor peripherally active.

Topic and activeness correlate in that the most easily processed sentences are those whose topic referents are active in the discourse (Lambrecht, 1994, p.165). According to Lambrecht, who defines the topic in the addressation sense, the ideal surface realization of the TA is an unaccented pronoun as in (5.5) and (5.6-5.8). Non-identifiable TAs result in barely acceptable sentences as the one in (5.9):

[1] There exist other hierarchies of salience which discriminate within what is labeled accessible (Prince, 1981; Gundel et al., 1993), but these finer distinctions are not relevant for us here.

(5.9) *?? [Eine Frau] ist [berühmt].*
A woman is famous.
'A woman is famous'.

The boundaries between the three categories are fuzzy, and attribution to one of the categories depends on several factors. Accessibility of a concept or a proposition, for example, can be provided by a direct or by an indirect reference, or by the activation of a concept related to the current one.

5.2 Sentence Topics and Local Coherence

As proposed in the previous section, there are two candidate positions for the TA: the VF and the position right after the verb, at the beginning of the MF. As the VF also seems to be a good position for the TF, the question arises, which of the two topics should be in the VF so that the discourse is as coherent as possible. To answer this question, we formulate the following hypothesis:

Hypothesis The VF, being a cognitively prominent position, has two major[2] functions:

1. The VF is the place for the TF, i.e., for the topic which characterizes the proposition expressed in the sentence as a whole and for other elements which help to position the proposition in the hearer's consciousness.
2. The VF is also the place for inactive TAs, i.e., TAs which have to be established: they should be placed in the VF so that the hearer/reader knows to which address the proposition should be attached.

TAs have their preferred position right after the verb, at the beginning of the MF, provided that they are active in the mind of the hearer/reader. In this case they should be pronominalized and there is no need for them to occupy the VF.

The sentences in (5.10-5.11) serve as an illustration to our hypothesis:

(5.10) *[Nach dem Studium] ist [er nach England] umgezogen.*
After the studies has he to England moved.
'After his studies, he moved to England'.

(5.11) *[Max Planck] wurde [am 23. April 1858 in Kiel geboren].*
Max Planck was on the 23th April 1858 in Kiel born.
'Max Planck was born on the 23th of April in 1858 in Kiel.

[2] Other elements, such as contrastive topics, are encountered in the VF as well but considerably less frequently.

In (5.10), *er (he)* is used to refer to the person the sentence is about, i.e., it constitutes the TA. The TA is apparently active due to the preceding context which is signalized by the pronominalized reference. The TF in (5.10) is a temporal expression which occupies the VF. (5.11) is the opening sentence of the biography of Max Planck, and *Max Planck* is the TA there. In contrast to (5.10), the referent of *Max Planck* is yet inactive, and therefore, the reference is a full NP which is placed in the VF. The temporal expression *am 23. April* present in this sentence is thus not placed there. One might object that temporal expressions do not function as frame topics with the verb *geboren werden (to be born)* but rather as its arguments, and that the example is not quite correct. However, further in the same biography[3] the following sentence can be found:

(5.12) *[Im Dezember] wurde [sein dritter Sohn, Hermann, geboren].*
In the december was his third son Hermann born.
'In december his third son Hermann was born'.

The TA of (5.12), as well as of practically all the sentences in his biography, is Max Planck and not his son. The TA is active, and the temporal expression may occupy the VF.

Thus, the rules we formulated resolve potential conflicts concerning the location of the TA and the TF. In the following we will show that these rules also ensure a more coherent discourse than a strategy of placing the subject or the most active entity in the VF. Note that our hypothesis agrees with the views of both Lambrecht (1994) and Frey (2004). As Lambrecht points out, sentence initial position is the ideal place for the TA as long as the topichood of an entity needs to be established by a lexical expression. When this role is already established, the preferred topic expression is an unaccented pronoun whose position is no longer functionally relevant:

> "From my characterization of the preferred topic expression as an unaccented pronominal argument, whose function is to express the grammatical and semantic role played by a pragmatically ALREADY ESTABLISHED topic referent in a clause it follows that the position of such a pronominal expression is functionally speaking IRRELEVANT. Once a topic referent is pragmatically established, i.e. once the function of the topic expression is no longer to ANNOUNCE the topic referent but to mark its role as an argument in a proposition, there is no longer any functional reason for the topic to appear at the beginning of the sentence." (Lambrecht, 1994, p. 201)

Further Lambrecht (1994) states the following:

[3] Wikipedia article, checked in March 2009.

> "For the preferred-topic expression it is functionally speaking more important to be in close association with the predicate than to appear in sentence-initial position [...] Unaccented pronominal topics therefore tend to occur in or near the position in which the verb itself occurs, i.e., towards the beginning of the sentence in verb-initial or verb-second languages [...]" (Lambrecht, 1994, p. 201)

We assume that in German there is a preferred TA position, namely at the beginning of the MF, right after the verb. This becomes particularly apparent in sentences with sentential adverbials. In sentences lacking such an adverbial it still remains the preferred position given that it is active and that there is a suitable element for the VF, e.g., a TF.

5.2.1 Conditions on Topichood

The only requirement for an expression to become the TA is the identifiability of the referent. In particular, this constraint predicts infelicity of sentences such as the sentence (5.9) where the topic is not identifiable. Concerning the TF, it is much harder to list all the necessary conditions for an expression to have the frame-setting function. We suggest that **accessibility** is a criterion for frame-settinghood. By placing an accessible constituent in the VF the speaker helps the hearer link the new information conveyed by the sentence to the representation he has already built in his mind. This very general formulation unifies such new information as temporal or locative expressions with information accessible due to the preceding context or the extralinguistic situation. Temporal expressions – absolute, *am 24. Mai 1900 (on the 24th of May)*, or relative, *im gleichen Jahr (in the same year)* – belong to this group because of the accessibility of the time scale. Well-known locations (e.g., *Berlin*, *Sankt Petersburg*) and other familiar named entities (e.g., *ETH Zürich*, *IBM*) are expected to be as readily accessible as temporal expressions. They also set a temporal/locational frame for the sentence. Entities directly or indirectly accessible due to the preceding context, e.g., repeated mentions, inferentially accessible constituents (bridging anaphora) and other anaphoric elements are also found acting as TF:

(5.13) *[Nach Abitur* *in Gammelholm 1903] ging [Niels Bohr*
After graduating from high-school in Gammelholm 1903 went Niels Bohr
zu der Universität Kopenhagen].
to the University Copenhagen.
'After graduating from high school in Gammelholm in 1903, Niels Bohr went to the Universtity of Copenhagen.'

(5.14) *[Sein Studium] schloss [er mit einer Doktorarbeit im Jahr 1911] ab.*
His studies finished he with a dissertation in the year 1911.
'He finished his studies with a thesis in 1911.'

In (5.14), *sein Studium (his studies)* has not been explicitly mentioned in the preceding sentence but is easily accessible because the domain of education has just been activated.

Another class of accessible constituents consists of discourse connectives. They establish a relation between the proposition expressed in the current sentence and propositions expressed earlier in the discourse. *So (so), anschliessend (finally), damit (in this way)* are examples of such connectives but not subordinate conjunctions such as *weil (because), obwohl (although)* which conjoin clauses. Proadverbials, e.g., *dabei (in doing so), darüber (about that)*, are also included in this group. The reader might get the impression that it is inconsistent to unify such diverse phenomena as discourse connectives and noun phrases. However, we follow Webber et al. (2003) and consider discourse adverbials anaphors. From this point of view the fact that an adverbial connective and an inferrable NP are both treated as accessible elements, should not be surprising, because both of them are anaphoric.

5.2.2 Formalization

Unlike other approaches which investigate the relation between information structure and word order (Rambow, 1993; Hoffman, 1998), our scope is not limited to noun phrases only, but includes also adverbs and discourse connectives. Because of that we did not choose such a well-established framework as Centering (Grosz et al., 1995; Prince, 1999) for formalization. The modifications needed for such a formalization would require extending the notion of the (backward/forward-looking) center not just to constituents other than NPs but also to propositions, and would lead to a loss of conceptual simplicity of this framework.

Based on the preceding discussion, the following implications can be drawn for generating locally coherent text (formalized in Fig. 5.1). The first step is the identification of the topic of the sentence (Line 2), which is the address for new information. In case the TA has not been established yet, it is referred to with a full NP and placed in the VF (Lines 13-14). If it is already established and active, then framing elements are found and the best candidate is placed in the VF (Line 7). The pronominalized TA is placed next to the verb (Line 8), at the beginning of the MF (MF-i). If there is no TF, then the TA is placed in the VF (Line 10).

We hypothesize that this strategy provides smoother transitions than reserving the VF for the TA in all cases. Sections 5.3 and 5.4 provide evidence from different sources which confirm our hypothesis; Section 5.5 presents a way of automatically choosing the best candidate (either the TA or the TF, Lines 7, 10 and 14) for the VF using a machine learning method.

5.3 Corpus Study

Automatic topic identification is a difficult task, even if a discourse-status-based definition of the topic is adopted (Hajičová et al., 1995; Postolache et al., 2005). Given that we investigate a

function FILL-VF(c, VF, MF-i)

```
1:  ta ← FIND-TA(c)
2:  tf ← FIND-TF(c)
3:  if IS-ESTABLISHED(ta) then
4:      ta ← PRONOMINALIZE(ta)
5:      if tf ≠ null then
6:          VF ← tf
7:          MF-i ← ta
8:      else
9:          VF ← ta
10:     end if
11: else
12:     ta ← FULL-NP(ta)
13:     VF ← ta
14: end if
```

Figure 5.1: Algorithm for filling the VF of a clause

corpus of biographies, WikiBiography (Sec. 2.1.2), we may reasonably assume that whenever the biographee is mentioned in a sentence, it is he/she who the sentence is about, i.e., he/she is the TA of the sentence. Indeed, it is this person the whole article is about and it is this person we learn new information about while reading the text. Thus, the majority of sentences has a TA and it can be identified straightforwardly. To measure the degree of activeness of the TA we look at the form of the expression used. We assume that pronominalization indicates active TAs and that full NPs are used for accessible but inactive TAs. The former assumption is generally safe whereas the latter does not hold universally. It may well be that a full NP is used although the TA is active. For example, such cases have been attested when there is a shift in the situation (see the remarks in Gundel (1998), p.196).

We searched the corpus for sentences which have the TA – the ones mentioning the biographee – and looked at where the TA is placed in those sentences. We analyzed 1,166 biographies with an average length of 17 sentences, 19,352 sentences in total. Approx. 12,000 sentences mention the biographee (with the name or with a personal pronoun) and hence were of interest to us. Whenever such a sentence opens a new section in an article, we assume that the TA should be explicitly established, therefore a concrete reference to the person is needed and the referring expression should be placed in the VF, no matter what its syntactic function is. Whenever a sentence is preceded by one or several sentences which already are about the biographee, we assume the referent to be active in the mind of the reader. In such a case a

	VF	MF-i	MF-r	total
pronoun	27%	70%	3%	7,540
name	64%	31%	5%	4,477

Table 5.1: Distribution of TAs according to their position and form

pronominal reference should be used, and the preferred position for it is at the beginning of the MF. The sentences (5.15-5.16) and (5.17-5.18) illustrate this point:

(5.15) *Familie und frühe Jahre.*
Family and early years.
'Family and early years'

(5.16) *[Marie Curie] wurde [am 7. November 1867 als Maria Salomea*
Marie Curie was on 7th November 1867 as Maria Salomea
Sklodowska in Warschau] geboren.
Sklodowska in Warsaw born.
'Marie Curie was born in Warsaw on the 7th of November 1867 as Maria Salomea Sklodowska.'

(5.17) *[Zusammen mit ihrem Mann Pierre Curie und dem Physiker Antoine*
Together with her husband Pierre Curie and the physicist Antoine
Henri Becquerel] erhielt [sie 1903 den Nobelpreis für Physik].
Henri Becquerel received she 1903 the Nobel prize in physics.
'Together with her husband Pierre Curie and the physicist Antoine Henri Becquerel, she received the Nobel prize in physics in 1903.'

(5.18) *[Acht Jahre später] wurde [ihr der Nobelpreis für Chemie] verliehen.*
Eight years later was her the Nobel prize in chemistry given.
'Eight years later, the Nobel prize in chemistry was given to her.'

Following the title, (5.16) is the opening sentence of the biography of Marie Curie. The TA is established by placing the full name reference in the VF. Pronominalization and placing the constituent in the MF are deprecated. In (5.17-5.18) the situation is different: the biographee is already active, and a better candidate for the VF of (5.18) is available – a temporal expression. This expression sets the temporal frame for the sentence and thus opens it.

Our hypothesis predicts that most examples in the corpus would have an active TA in the beginning of the MF and an inactive one in the VF. Table 5.1 shows the distribution of the expressions referring to the biographee (pronominal and non-pronominal), i.e., TAs, with respect to their activeness (pronoun vs. name) and to their position in a sentence. The possible positions are the VF, the initial position of the MF (MF-i), and the remainder of the MF (MF-r).

	person	temp.expr.	conn.	unclass. NE	loc.	other	total
VF	26%	19%	5%	2%	1%	48%	19,352

Table 5.2: Constituents found in the VF

The results clearly indicate that the VF is not a preferred position for pronouns, whereas non-pronominal references appear in the VF about twice as often as in the MF-i. The pronouns, active TAs, are found in the MF-i position two and a half times more often than in the VF. Only an insignificant portion of TAs occur in the rest of the MF. Moreover, some of these cases result from the errors of the tagger or the parser, which incorrectly split the constituent in the VF into two and thus "shifted" the pronoun in the MF-i one position to the right.

As we have pointed out in the beginning of this section, a non-pronominal reference does not necessarily imply inactiveness of the referent. Therefore, names may also occur in the MF-i. A manual investigation of a sample of random sentences demonstrated that pronouns in the VF are usually found in sentences which lack a suitable TF. This explains why there are some pronouns in the VF and names in the MF.

To see whether the VF is indeed occupied by TFs, we checked which kind of constituents appears there. Table 5.2 presents the results. Approximately half of the constituents could not be classified automatically, but temporal expressions, connectives, and locations, which we consider good candidates for the TF, account for a considerable number of cases. Temporal expressions frequently occur in the VF because they represent the TF for these sentences. Anaphoric connectives and locations are less frequent but still account for about one thousand sentences. Persons, some of them being TAs, appear in the VF quite frequently. Since the corpus analysis gives an indication of a certain tendency only, it is of interest to see whether this tendency correlates with more coherent discourses. In order to find out this we performed an experiment with human judges.

5.4 Experiment with Native Speakers

We performed an experiment with human judges, all native speakers of German, who were presented with 24 short text fragments from our corpus. Each fragment had two possible continuations which were identical in all aspects but the word order. The order of the two alternative sentences as well as the order of the fragments was generated randomly. The judges were asked to choose from the two variants the one which continues the preceding text (one or two sentences) in the most fluent way or choose nothing in case both continuations sound equally fluent. The following sentences constitute a test fragment:

(5.19) *[Nach seiner Kriegsteilnahme am Ersten Weltkrieg] folgte [er*
After his war participation in the First World War accepted he
Berufungen nach Jena, Stuttgart, Breslau und Zürich].
appointments to Jena, Stuttgart, Wroclaw and Zürich.
'Having taken part in the First World War, he accepted appointments from Jena, Stuttgart, Wroclaw and Zürich.'

(5.20) *[Dort] belegte [er den Lehrstuhl für Theoretische Physik, den vor*
There occupied he the chair for theoretical physics, which before
ihm bereits Albert Einstein und Max von Laue inne hatten].
him already Albert Einstein and Max von Laue hold.

(5.21) *[Er] belegte [dort den Lehrstuhl für Theoretische Physik, den vor*
He occupied there the chair for theoretical physics, which before
ihm bereits Albert Einstein und Max von Laue inne hatten].
him already Albert Einstein and Max von Laue hold.
'He occupied there the chair of theoretical physics, which was before him hold by Albert Einstein and Max von Laue.'

Sentences (5.20) and (5.21), preceded by (5.19), have the same propositional content and differ only in what is placed in the beginning of the sentence – the proadverbial *dort (there)* or the personal pronoun *er (he)*. According to our hypothesis, the judges should choose (5.20) more often than (5.21) because the TA is active (*er*) and placed in the MF-i, and the TF (*dort*) occupies the VF.

The purpose of the experiment was mainly two-fold:

- to check whether in cases where topic establishing is necessary (e.g., (5.16)), the VF is the preferable position for the topic;
- whether an established topic occupying the VF makes the transition to the sentence smoother, or there are better candidates for this position (see (5.19-5.21)).

Nineteen human judges – ten female and nine male participants – took part in the experiment. The statistical significance of our results was computed using the χ^2 test. The preference for a certain variant was counted as significant on the $p = 0.01$ level or below if it was chosen by at least sixteen judges.

5.4.1 Topic-Establishing Sentences

We selected three section-initial sentences mentioning the biographee because such sentences open a new discourse topic (this is explicitly marked by using section titles) and therefore require non-pronominal reference to the person. Three pairs, each consisting of a sentence

	TA in VF
significance	– + +
number of judges	12 16 18

Table 5.3: Results for the topic-establishing sentences

and of a propositionally equivalent variant of it, were presented to the judges. The sentence (5.16) is one of those fragments. In these three fragments the judges had a choice of what to place in the VF: (i) an absolute temporal expression, (ii) an NP with a reference to a previously mentioned and therefore accessible referent, and (iii) an inferentially accessible NP or a name reference to the biographee (i-iii). In all three cases the biographee was preferred over other candidates for the VF position, and in two of the cases the difference was significant. Note that this finding alone is in accordance with the well-known correlation between topics, subjects and sentence initial positions and does not favor our hypothesis more than many others.

A plus (+) in Tables 5.3 and 5.4 stands for cases where the difference in preferences is significant on the $p = 0.01$ level, a circle ($\circ$) for significance on the $p = 0.05$ level, a minus (–) for non-significant preference. The number of judges whose choice is in accordance with the hypothesis is given below each significance label.

5.4.2 Sentences with Established Topic

The second part of the experiment concerned sentences where the biographee is established as the TA due to the immediately preceding context (such as (5.17-5.18) and (5.19-5.20)). From the 19 test pairs of this kind, 17 contained a pronominal reference, and in two other pairs the biographee was referred to with the last name. For these examples, constituents of the following kinds were supposed to be better candidates for the VF than active TAs: inferrable constituents (five fragments), temporal expressions (four fragments), discourse connectives (five fragments), or proadverbials (five fragments). Here we distinguish between connectives which have a distinct semantic meaning (e.g., temporal or additive) – these are labeled as discourse connectives – and proadverbials (*davon – from that, darüber – about that*) whose meaning is usually context-dependent.

Syntactic function was expected to play a minor role for the choice of the best constituent for the VF. This parameter was set in favor of the active referent: in all sentences the syntactic role of the biographee was the subject.

For every pair it turned out that the majority of judges preferred accessible constituents over active subjects. In five cases, the judges preferred the modified version over the original sentence (i.e., the sentence from the Wikipedia article) because they found the modified fragment sounds more fluent (Table 5.4). Interestingly, for both examples with a non-pronominal

	inferrable	temp.expr.	connective	proadverbial	total
pronoun	– + + +	– ○ + +	– ○ ○ +	– – – – +	17
	10 16 17 18	13 14 16 18	10 15 15 16	10 10 11 12 17	
name	+		○		2
	16		15		

Table 5.4: Results for the sentences with established topics

reference to the biographee the connective as well as the accessible constituent were preferred significantly more often. This brings us to the conclusion that, for a fluent transition, the established topic should not be placed in the VF no matter what its surface or syntactic realization is.

The last two test sentence pairs let the reader choose between, first, a temporal expression and an accessible constituent; second, a temporal expression and a proadverbial. For the former case, no difference in preferences was found; for the latter, the proadverbial was chosen significantly more often than the temporal expression. Obviously, in order to rank candidates for VFs of different kinds (e.g., to claim that temporal expressions are better than bridging anaphoric NPs) more subtle experiments need to be performed: Form of the expression, semantics of connectives, and degree of accessibility should be taken into account. So far, it can only be stated that concerning candidates for the VF, the established TA follows any of the ones listed in Section 5.2, i.e., it follows the TF.

5.5 Generation Experiment

In this section we present a possible implementation of our findings. Following the algorithm in Figure 5.1, we focus on the step of determining the VF constituent (Lines 7, 10 and 14) and show how this can be done automatically. This can also be seen as a first step in the constituent ordering process, the other step being the ordering of the remaining constituents in the MF. Here, splitting the task in two is judicious as the reasons which promote a constituent to the VF differ from those that determine its position in the MF. Being an ideal candidate for the VF, the TF is determined by its semantic and discourse properties whereas the order in the MF seems to primarily depend on syntactic roles of constituents (Kempen & Harbusch, 2004c). The TA is to be placed in the VF if it is inactive or if there is no better candidate. It is of interest to see whether the best candidate for the VF can be found automatically by a supervised learner.

For an illustration, consider the sentence from the example in (5.20) again together with its dependency structure. For our purposes we may ignore the structure dependent on the

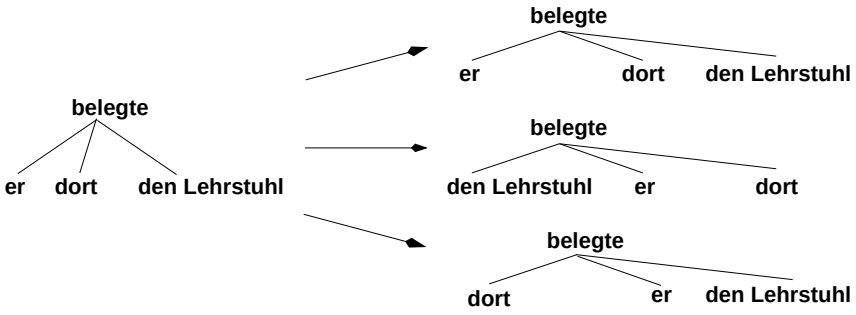


Figure 5.2: Essential part of the example in (5.20)

training	development	test
14,324	3,344	1,683

Table 5.5: Size of the data sets in sentences

accusative object *Lehrstuhl* and consider only the nodes dependent on the root verb (Figure 5.2).

(5.22) [Er] belegte [dort den Lehrstuhl für Theoretische Physik, den vor ihm bereits Albert Einstein und Max von Laue inne hatten].

In this example there are three candidates which can occupy the VF because there are three constituents dependent on the main verb. The task of the learner is to predict which of the three should be placed in the VF based on what it has learned from the training set of sentences. Two baseline algorithms are tested on this task. The first one picks a constituent randomly, the second one always selects the subject for the VF. Both of them result in an accuracy of approximately 30%.

5.5.1 Data

To obtain data necessary for training, we split approx. 19,000 sentences from the WikiBiography into the training, development and test sets and select parsed sentences which mention the biographee. We sort out clauses with only one constituent so that the number of candidate

2	3	4	5	6+
20%	35%	27%	12%	6%

Table 5.6: Proportion of sentences with certain length

constituents for a sentence ranged from two to eight with 3.5 constituents on average (Tables 5.5 and 5.6).

Using maximum entropy learning (Berger et al., 1996) which has been successfully applied to a number of NLP tasks including word order generation (Ratnaparkhi, 2000; Uchimoto et al., 2000), we train a binary classifier (OpenNLP[4]). Maximum entropy learners can cope with a large number of non-numerical features which is very important for our task. Trained on a large number of instances correctly classified as VF or MF, the classifier estimates the probabilities of each label for every testing instance. During testing phase, for every sentence, the constituent with the highest probability of being in the VF is selected for this position. The results are then evaluated against the source sentence.

5.5.2 Features

The three feature vectors for the three constituents in Figure 5.2 are presented in Table 5.7. We used the following features in our experiments:

DW: the lemma of the dependent word, i.e., the word immediately dependent on the verb;

VERB: the lexical part of the verb;

LEX: the lexical head of the dependent constituent (if different from the dependent word);

POS: part of speech of the dependent word;

SYNT: the syntactic function of the constituent;

DL: the 'weight' of the constituent – i.e., its depth in the dependency tree and length in words it covers. Since our learner treats all values as nominal, we discretized the depth and the length numeric values with a J48 classifier (Kohavi & Sahami, 1996). It turned out that there is an essential difference between depths greater than or equal to two and those smaller than two. The possible range of lengths was also split into two classes: lengths greater than or equal to three, and the rest. DL is a complex feature which describes the depth and the length simultaneously in order to overcome the learner's inability to deal with dependent features. The three possible values of DL are *ss*, *sl*, *ll*, where *s* and *l* stand for *small* and *large* numbers respectively. Note that the value *ls* is impossible since even the minimal depth of two assumes a length of at least three words.

CONN: whether the constituent is a connective;

SEM: whether the constituent is a named entity, temporal expression, or a person;

[4] `http://opennlp.sourceforge.net`

DW	VERB	LEX	POS	SYNT	DL	CONN	SEM	TA	RE
er	belegen	er	pper	subj	ss	no	pers	yes	pron
dort	belegen	dort	adv	adv	ss	no	–	–	–
lehrstuhl	belegen	lehrstuhl	nn	obja	ll	no	–	–	–
am	geboren	november	card	pp	ll	no	temp	–	abs

Table 5.7: Feature vectors for the constituents in Figure 5.2 and the temporal expression from the sentence (5.16)

RANDOM	30%
SUBJECT	30%
MAXENT	65%

Table 5.8: Accuracy of the two baselines and the classifier

TA: whether the constituent is the TA, i.e., whether it refers to the biographee;

RE: specifies the type of referring expression which can be either *pron* (i.e., pronoun) or *name* for the biographee, or *abs* (i.e., absolute) or *rel* (i.e., relative) for temporal expressions. The last line in Table 5.7 gives the values of the temporal expression from the sentence (5.16) – *am 7. November 1867*.

The first seven features are applicable to any candidate. Note that contextual information is not encoded in our features, and as a result inferrable constituents cannot be identified.

5.5.3 Results

We evaluated the performance of the classifier with **accuracy** – the ratio of correctly predicted VFs to the total number of test sentences. From about 1,700 test instances the algorithm made a mistake in 600 cases, which results in the accuracy of about 65% (Table 5.8). This is by far better than any of our baselines (30%).

5.5.4 Error Analysis

We inspected the first 100 errors. These can be grouped into different classes (Table 5.9). We found the cases with inferrables particularly hard to solve which is not surprising given that none of the features encodes inferrability. In 22 cases a pronominal reference to the biographee was chosen instead of a NP, PP or a subordinate clause (all three are labeled XP in the table) which were accessible due to the preceding context.

Wikipedia	MaxEnt	#
pron	temp	17
pron	conn	8
name	temp	11
XP	pron	22
parser error		12
contr. topic		5
equiv. cases		9

Table 5.9: Types of errors with their frequency

In 17 cases the algorithm preferred a temporal expression over a pronoun which occupied the VF in the original Wikipedia article. This counts as incorrect although, as the experiment described in Section 5.4 demonstrated, human judges find a text more coherent when there is a temporal expression and not a pronoun in the VF. Likewise, the fact that eight connectives were classified incorrectly does not imply that the generated order would make the text less coherent than the original. Apart from that, name references may have been used in sentences with established topics, which means that some of the eleven errors might not be serious ones.

Some errors were caused by the tagger or by the parser which failed at identifying the main verb and/or could not build the correct parse. In five cases there was a contrastive topic in the VF which the system in its present configuration cannot recognize. In nine cases the generated sentence was as good as the original one. The rest of the errors cannot be classified as one of the above. These are cases where the TF was accessible for different reasons. An example of such a situation is given below:

(5.23) *[Ihr Mann] war [der Physiker Pierre Curie].*
Her husband was the physicist Pierre Curie.
'Her husband was the physicist Pierre Curie.'

(5.24) *[Zusammen] erhielten [sie 1903 den Nobelpreis in Physik].*
Together received they 1903 the Nobel Prize in physics.
'Together they received the Nobel Prize in physics in 1903.'

The adverb in the VF of the sentence (5.24) is accessible as it refers to Marie and Pierre Curie. This anaphoricity can be detected neither by similarity features nor can it be labeled as accessible in advance the way it was done with connectives and proadverbials.

5.5.5 Conclusions

In this section we presented an experiment whose goal was to fill the VF automatically. Although the features we used do not model the information structure fully, the results we obtained are encouraging. They demonstrate that our system by far outperforms the 'subject-first' baseline. However, TA and TF identification are both difficult tasks which require a separate study across different genres. It should be noted that the former task is easily solvable on our corpus of biographees. Unlike that, the semantics of the TF is complex and requires features which could not be modeled with the annotation we had: e.g., the accessibility was encoded poorly.

5.6 Related Work

The question of how to select the best order from a number of grammatical ones is not new. For example, Kruijff et al. (2001) describe a system architecture which supports generating the appropriate word order for languages with syntactic or pragmatic constraints on it. Inspired by the findings of the Prague School (Sgall et al., 1986) and Systemic Functional Linguistics (Halliday, 1985), they focus on the role that information structure plays in constituent ordering. In their study, they consider English, Czech and German and demonstrate that, in each case, the word order can be determined by the so called **communicative dynamism** (Firbas, 1974) as well as by the language specific **systemic ordering** (Sgall et al., 1986). Generally, communicative dynamism prescribes that explicitly or implicitly given entities (termed context-bound) precede new information and systemic ordering describes the canonical order in a clause which in case of German corresponds to the following:

$$\text{Actor} \prec \text{Temporal} \prec \text{SpaceLocative} \prec \text{Means} \prec \text{Addressee} \prec \text{Patient} \prec \text{Source} \prec \text{Destination} \prec \text{Purpose}$$

Since in this theory discourse status determines the position of a constituent in the sentence – old information precedes new one – sentences which violate this principle are judged as infelicitous. However, our experiments have shown that discourse status alone does not explain frequently occurring patterns in our data. The interplay between discourse status and topicality explains German constituent ordering more adequately.

Kruijff et al. (2001) apply their algorithm to English and Czech software instruction manuals and note that it can also be applied to German. They concentrate on how to generate not just a grammatical but an acceptable ordering. Our goal lies even further as we aim at determining the ordering that makes the transition between sentences as smooth as possible, i.e.,

we want to find the best of all acceptable orders. In addition we extend context-boundedness to accessibility and treat context-bound NPs, temporal expressions and discourse connectives uniformly. Apart from that, the preferences reported in Section 5.4 can be explained neither in terms of communicative dynamism nor systemic ordering. We showed that a more salient actor is placed after a less salient temporal expression, which contradicts both of the scales.

Kruijff-Korbayová et al. (2002) build upon Kruijff et al. (2001). Similar to our approach, their algorithm recognizes the special role of the sentence-initial position which they reserve for the **theme** – the point of departure of the message. In this chapter we have shown that the notion of the theme alone is insufficient for solving the constituent ordering task in German.

Most models of local coherence utilize the Centering model (Grosz et al., 1995) and concern the choice of referring expression (see Poesio et al. (2004) for an overview) or suggest algorithms for ordering such discourse units as sentences or clauses (Barzilay et al., 2002; Lapata, 2003; Karamanis et al., 2004, inter alia). Since all these studies concern English, which demonstrates relatively rigid word order, the question of constituent ordering have not played an important role there.

5.7 Summary

In this chapter we considered the relation between the VF and local coherence and demonstrated that the perceived fluency of discourse depends on what is placed in the VF. Having adopted the views of Lambrecht (1994), Jacobs (2001) and Frey (2004) on the diverse functions of the sentence topic, we have suggested a solution to the generation of fluent transitions between German sentences and have demonstrated the importance of the sentence-initial position for local coherence.

Corpus investigation as well as experiments with human judges on constituent reordering confirmed our claims concerning the role of the VF. In most cases, it is either the establishing position for the addressation topic, or the position for accessible constituents – those which set the frame for the whole sentence. In line with our hypothesis, human judges find transitions between sentences smoother when the VF is occupied by the TF, and not by the TA, unless the TA status of a constituent has to be established.

The findings of this study as well as the initial experiments with automatic filling the VF provide a basis for our linearization algorithm which we introduce in the next chapter.

Chapter 6

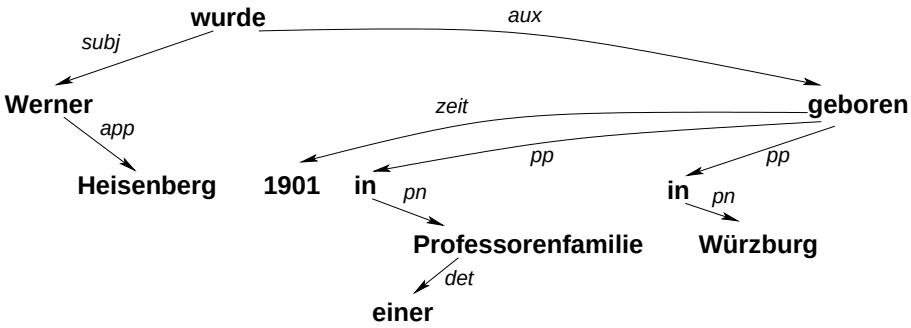
Dependency Tree Linearization

In this chapter we present our tree linearization algorithm which takes a dependency tree as input and outputs a string of words. What we are going to advocate here is a **combined approach** which distinguishes the task of ordering phrases dependent on the finite verb from the one of finding the best word order within those phrases. Consequently, the tasks are solved differently. The focus of this chapter is on German as this is the language for which deFuser has been developed and on which it has been tested so far. However, we will also show that our combined approach achieves good results for English. Before we explain how the combined method works (Sec. 6.3), we first familiarize the reader with the terminology used throughout this chapter (Sec. 6.1) and give an overview of related work on word order generation (Sec. 6.2). Sections 6.4 and 6.5 introduce the methods we implemented for ordering verb dependents and the words within them respectively. Both sections present the results of automatic evaluation.

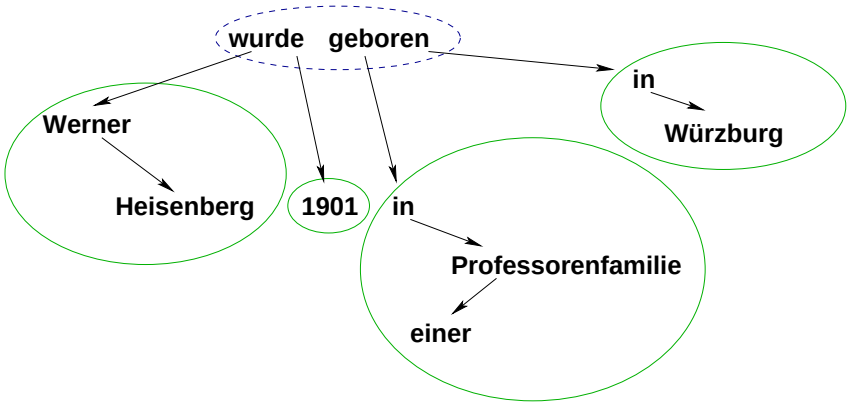
6.1 Terminology

The definitions we introduce in this section are illustrated with an example taken from CoCoBi[1] (6.1); Figure 6.1a shows the dependency tree of the sentence. Recall from Chapter 2 that the WCDG parser treats the finite verb of a compound verb as the root of the clause. The subject depends on the finite verb whereas other arguments depend on the lexical part of the compound verb. In our work and in the following discussion, in order to simplify the matter, we merge the verb parts into one node as is shown in Figure 6.1b. This is a fairly simple modification which eliminates the (for our purposes) irrelevant information from the representation.

[1] See file 10248.



(a) Dependency tree produced by the WCDG parser



(b) Dependency tree with a single verb node

Figure 6.1: Dependency tree of the sentence in (6.1)

(6.1) *Werner Heisenberg wurde 1901 in Würzburg in einer Professorenfamilie*
Werner Heisenberg was 1901 in Würzburg in a professor family
geboren.
born.
'Werner Heisenberg was born in 1901 in Würzburg in a professor family.'

Constituent is usually defined as a group of words or a single word which functions as a single unit in a hierarchical syntactic structure. We use this term to refer only to a subset of constituents which are dependent on a verb. In Figure 6.1b there are four constituents in our sense (encircled in Fig. 6.1b).

Head of constituent is defined as the highest word in the constituent which plays the same grammatical role as the whole constituent. In Figure 6.1b the heads are *Werner*, *1901*, *in* and *in*. Alternatively it can be defined as a word whose parent does not belong to the constituent.

Subtree of constituent is the subtree of the dependency tree rooted in the constituent's head.

Lexical head of constituent is the highest open-class node in the subtree of constituent. For prepositional phrases, it is the noun (*Professorenfamilie* and *Würzburg* in Fig. 6.1b).

Parent of constituent is the parent of its head, i.e., the verb (*wurde geboren*).

Length of constituent is the total number of words covered by the constituent. In Figure 6.1b the lengths are two, one, three and two, respectively.

Depth of constituent is the maximum depth of the subtree of the constituent. In our example these are one, zero, two and one.

Constituent linearization refers to the task or process of ordering words within constituents. For example, the correct linearization of the third constituent in Figure 6.1b results in *in einer Professorenfamilie*.

Constituent ordering refers to the task or process of finding the order among constituents. In Figure 6.1b the task is to find the correct order of the four constituents.

6.2 Previous Work on Word Order Generation

6.2.1 Trigram LM Based Approaches

Trigram models are easy to build and use, and it has been shown that more sophisticated n-gram models (e.g., with higher n, complex smoothing techniques, skipping, clustering or

caching) are often not worth the implementation effort due to data sparseness and other issues (Goodman, 2001). This explains the popularity of trigram LMs in a variety of NLP tasks (Jurafsky & Martin, 2008), in particular, in tree linearization where they have become the de facto standard tree linearization tool in accordance with the 'overgenerate and rank' principle. According to this principle, given a syntactic tree, one needs to consider all possible linearizations and then choose the one with the lowest entropy. This straightforward solution to surface realization was first proposed by Langkilde & Knight (1998) and can be applied to basically any system which at some point needs to get a sentence from an underlying representation. Within sentence fusion architectures this approach has been undertaken by Barzilay & McKeown (2005) and Marsi & Krahmer (2005) for English and Dutch respectively. In both projects dependency trees are provided as input and the ranking is determined with a trigram LM trained on a large text corpus.

Usually, language models are expected to work better on languages with rigid word order. This is not surprising given that the basic unit of language modeling is *n*-gram, that is, a sequence of *n* tokens. The idea is that a word can be predicted from its preceding context, namely, from *n-1* previous words. Naturally, for languages with a free word order it is harder to make such a prediction simply because there is a greater variety of possible trigrams. On the scale of "word order freedom" of a given language, English belongs to the group of rigid word order languages (Givón, 2001, inter alia), and therefore LMs built on English data are supposed to work sufficiently well. The situation is different with, e.g., Dutch, German or Slavic languages which have a richer morphological marking and are considered to be languages with relatively free word orders. Marsi & Krahmer (2005), who use a trigram LM trained on 250M words from the Twente Newscorpus to rank tree linearization variants in Dutch, observe that

> "the ranking was often inadequate, showing ungrammatical variants at the top and grammatical variants in the lower regions." (Marsi & Krahmer, 2005, p. 5).

It should be noted that long-distance dependencies exist in any language – English being no exception – and they are impossible to recognize with an *n*-gram model where *n* is not really big. Unfortunately, greater values of *n* require a considerably larger corpus to cope with the data sparseness problem. This tradeoff between the quality of the *n*-gram model and the amount of training data needed has been acknowledged in many studies (Jurafsky & Martin, 2008). The situation is very likely to improve in the future given that such resources as, for example, Google's N-gram Corpus[2], which includes five-grams, are already available. Presently, usage of large LMs is hindered because they require computational capacities far

[2] Available since September 2006 from
`http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2006T13`.

beyond those which conventional computers can offer. All this being said, the most common solution at the moment is to use a trigram model with smoothing (e.g., Good-Turing, Witten-Bell or Katz's backing-off (Manning & Schütze, 1999)).

6.2.2 Other Approaches

When using a trigram LM for surface realization, one ignores linguistic information which might be available in the representation: e.g., syntactic or part-of-speech information. This is unfortunate given that it provides additional clues to what the right word order should be. Therefore, a number of methods have been developed which aim at utilizing linguistic knowledge in determining the word order. Below we give a short overview of such methods.

Uszkoreit (1987) addresses the task from a grammar-based perspective within Generalized Phrase Structure Grammar (GPSG) as developed by Gazdar et al. (1985). One of the goals of his study is to provide a description of the constraints on the order of constituents in German. Among other phenomena, Uszkoreit (1987) considers the differences in the constituent order between the main and subordinate clauses, topicalized constituents in main clauses, the order of verb complements and adjuncts. He suggests weighted constraints on precedence such as

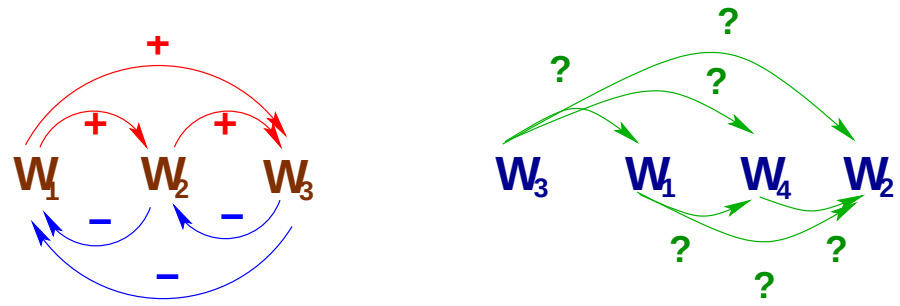
> [+NOM] $\prec$ [+ACC] (i.e., constituents in nominative case precede those in accusative case),
>
> [+PRO] $\prec$ [–PRO] (i.e., pronominalized constituents precede non-pronominalized ones),
>
> [–FOCUS] $\prec$ [+FOCUS] (i.e., focused constituents follow those which are not in focus).

Apart from the descriptive goal, the study of Uszkoreit (1987) has a theoretical goal of investigating whether the syntax of German can be adequately described within GPSG and proposes a set of phrase structure rules for a fragment of German syntax. Suggesting directions for future research, Uszkoreit (1987) points out that the interaction of pronominalization, definiteness, and discourse role assignment is complex and yet to be understood.

The studies of Kruijff et al. (2001) and Kruijff-Korbayová et al. (2002) (mentioned in Sec. 5.6) present an architecture for generating appropriate word order for different languages based on the ideas developed by Halliday (1985) and within the Prague School (Sgall et al., 1986). Unfortunately, they did not implement their algorithm, and it is hard to judge how well the system would perform on real data.

Harbusch et al. (2006) present a generation workbench for German and Dutch, which has the goal of producing not the most appropriate constituent order, but all grammatical ones. They also do not provide experimental results.

The work of Uchimoto et al. (2000) is done on Japanese - a free word order language. Uchimoto et al. (2000) determine the order of phrasal units dependent on one word. Their approach is similar to ours in that they aim at regenerating the original order from a dependency



(a) Generating training instances from the data

(b) Estimating the probability of the order

Figure 6.2: The training and testing phases of the system of Uchimoto et al. (2000)

parse. However, they regenerate the order of modifiers for all nodes and not only for the verb. Using a maximum entropy framework, they choose the most probable order from the set of all permutations of n words with the following formula:

$$\begin{aligned} P(1|h) &= P(\{W_{i,i+j} = 1 | 1 \leq i \leq n-1, 1 \leq j \leq n-i\}|h) \\ &\approx \prod_{i=1}^{n-1} \prod_{j=1}^{n-i} P(W_{i,i+j} = 1 | h_{i,i+j}) \\ &= \prod_{i=1}^{n-1} \prod_{j=1}^{n-i} P_{ME}(1 | h_{i,i+j}) \end{aligned} \tag{6.2}$$

For each permutation, for every pair of words, they multiply the probability of them being in the correct order given the history h (see Fig. 6.2). Only reference orders are assumed to be correct. Random variable $W_{i,i+j}$ is 1 if word w_i precedes w_{i+j} in the reference sentence, or zero otherwise. Figure 6.2a gives an example where from a sequence of three words three positive and three negative instances are created from the total of six possible pairs. Figure 6.2b demonstrates which probabilities should be computed to estimate the probability of the sequence of four words ($4 \times 3/2$, i.e., six). The features they use are akin to those which play a role in determining German word order. We use their approach as a non-trivial baseline in our study (see Sec. 6.4.3.4).

Ringger et al. (2004) aim at regenerating the order of constituents as well as the order within them for German and French technical manuals. Utilizing syntactic, semantic, subcategorization and length features, they test several statistical models to find the order which maximizes the probability of an ordered tree. Using "Markov grammars" as the starting point and conditioning on the syntactic category only, they expand a non-terminal node C by pre-

dicting its daughters from left to right:

$$P(C|h) = \prod_{i=1}^{n} P(d_i|d_{i-1}, ..., d_{i-j}, c, h) \tag{6.3}$$

Here, c is the syntactic category of C, d and h are the syntactic categories of C's daughters and the daughter which is the head of C respectively. In their simplest system, whose performance is only 2.5% worse than the performance of the best one, they condition on both syntactic categories and semantic relations (ψ) according to the following formula:

$$P(C|h) = \prod_{i=1}^{n} \left[\begin{array}{l} P(\psi_i|d_{i-1}, \psi_{i-1}, ...d_{i-j}, \psi_{i-j}, c, h) \\ \times P(d_i|\psi_i, d_{i-1}, \psi_{i-1}..., d_{i-j}, \psi_{i-j}, c, h) \end{array} \right] \tag{6.4}$$

Although they test their system on German data, it is hard to compare their results to ours directly. First, the metric they use does not describe the performance appropriately (see Section 6.4.4.1). Second, while the word order within NPs and PPs as well as the verb position are prescribed by the grammar to a large extent, the constituents can theoretically be ordered in any way. Thus, by generating the order for every non-terminal node, they combine two tasks of different complexity and mix the results of the more difficult task with those of the easier one.

More recently, Velldal & Oepen (2006) compare three methods for surface realization ranking in English. Given a representation in the form of Minimum Recursion Semantics, they generate a ranking of possible realizations (i) with a trigram LM; (ii) with structural features in a maximum entropy model; and (iii) with the same features using support vector machines. They show that a combination of structural features with the LM in a maximum entropy model performs best.

Cahill et al. (2007) use a log-linear model to rank surface realizations in German. The strings are derived from a corpus of F-structures by a Lexical Functional Grammar. Their results show that a combination of LM scores with linguistic features brings a significant improvement over rankings which are based solely on a LM.

6.3 Combined Approach to Tree Linearization

The motivation for our approach is based on the premise that the 'freedom' of word order concerns the order of constituents in the first place because word order **within** constituents is considerably more rigid than the order **of** constituents. Therefore, determining their order is a considerably more challenging task and requires more linguistic knowledge than the one of linearizing noun or prepositional phrases. As an illustration to the point just made, consider the following examples:

(6.5) *[Mit nur 26 Jahren] wurde [Heisenberg] [als Professor an die Universität Leipzig] berufen.*
With only 26 years was Heisenberg as professor at the university of Leipzig called.
'With only 26 years old Heisenberg was called as a professor at the university of Leipzig.'

(6.6) [Heisenberg] wurde [mit nur 26 Jahren] [als Professor an die Universität Leipzig] berufen.

(6.7) [Heisenberg] wurde [als Professor an die Universität Leipzig] [mit nur 26 Jahren] berufen.

(6.8) [Als Professor an die Universität Leipzig] wurde [Heisenberg] [mit nur 26 Jahren] berufen.

The order of constituents – these are marked with square brackets in all the examples – can vary considerably in German: all of the variants listed in (6.6-6.8) are grammatical. Unlike that, the order within each of the phrases in brackets is the only possible order because the grammar puts hard constraint on noun modifiers – determiners always precede their noun heads as well as adjectives; likewise prepositional phrases modifying nouns always follow them. Such rules cannot be formulated for verb modifiers (at least not in German). Generally speaking, subjects tend to precede direct objects but this is definitely not a rule, as we have seen in the previous Chapter. Similarly, the order of constituents dependent on the verb may vary also in English, albeit to a lesser extent. For example, both sentences below are grammatical:

(6.9) [With only 20 years old] [she] had [one of the most powerful and eclectic voices in the music business].

(6.10) [She] had [one of the most powerful and eclectic voices in the music business] [with only 20 years old].

Based on this observation, our combined approach does not linearize the whole tree at once. Instead, it recursively finds the best order for each clause. Within the clause, it first linearizes the constituents' subtrees and then determines the order of the constituents. A trigram LM-based method is used for the simpler task of constituent linearization; a richer representation is utilized to solve the more difficult task of constituent ordering. The recursive linearization algorithm is presented in pseudocode in Figure 6.3. The LINEARIZE(*verbNode*) function takes a tree rooted in a finite verb as input. For every child c of the verb, it linearizes the constituent rooted in this child with the BUILD-CONSTITUENT(*c*) function. Once all constituents are collected, their order is determined with ORDER-CONSTITUENTS(*constituents*). Then the

function LINEARIZE($verb$) **returns** String

List $cons \leftarrow \{\}$
for all $c \in verb$.`children`() **do**
$cons$.`add`(BUILD-CONSTITUENT(c))
end for
$cons \leftarrow$ORDER-CONSTITUENTS($cons$)
INSERT-VERB($cons, verb$)
return $cons$.`toString`()

function BUILD-CONSTITUENT($node$) **returns** Constituent

if IS-VERB($node$) **then**
return Constituent(LINEARIZE($node$))
else
List $orders \leftarrow$ GET-POSSIBLE-ORDERS($node$)
return Constituent(GET-BEST-WITH-LM($orders$))
end if

Figure 6.3: Tree linearization algorithm

finite verb is inserted right after the first constituent, if the input tree corresponds to a main clause, or added in the end, otherwise. Non-finite parts of a compound verb are placed at the end of the clause. The complete algorithm implemented in Java can be found in the software package available online[3]. Section 6.4 presents several methods of how ORDER-CONSTITUENTS(*cons*) can be implemented. Section 6.5 describes the recursive algorithm of GET-POSSIBLE-ORDERS(*node*).

6.4 Constituent Order Generation

In this section we first outline the factors which have been claimed to have an influence on the constituent order in German and then describe the methods we have implemented. In particular, we introduce our **two-step** method (Sec. 6.4.3.6) which, in a nutshell, first selects a constituent which would be the best starting point for the sentence (i.e., it fills the VF), and then efficiently orders the remaining constituents. Our empirical results demonstrate that such a separated treatment is beneficial and provides a significant improvement in performance.

[3] See `http://www.eml-research.de/~filippova`

6.4.1 Relevant Factors as Found in Previous Studies

In the previous chapter we demonstrated that several factors play a role in determining the order of (German) constituents. Below we present the findings of mostly corpus and psycholinguistic studies which investigated the influence of syntactic, semantic, discourse and surface features on word order. Note that most of these studies concern the order of verb arguments only. Little has been said so far about how non-arguments should be ordered.

Hawkins (1992) measures Immediate Constituent to Word Ratio (ICR) – the ratio of the constituent number divided over the position of word – and predicts that from sentences with a higher ICR (averaged over words) should be preferred over those with a lower ICR. This is based on the assumption that "lighter" constituents are easier to process for humans and thus tend to appear earlier in a clause. In particular, the theory predicts that pronouns precede full NPs which holds in German as in many other languages. However, it judges [+PRO+ACC] $\prec$ [+PRO+DAT] and [+PRO+DAT] $\prec$ [+PRO+ACC] as equally good while only the former is a grammatical order in German.

"Heavy NP shift" is a related phenomenon which predicts that grammatically complex NPs (e.g., a long NP modified by a relative clause) appear at a position right to their canonical position (Ross, 1967). This term emerged within transformational grammar but has been adopted by linguists who do not belong to this tradition.

Kurz (2000) investigates the role that the length of the constituent plays in determining its position in a clause. She considers six verbs and restricts the investigation to non-pronominal NPs in the MF in NEGRA[4] and in the Frankfurter Rundschau Corpus[5]. The results demonstrate that there is no general pattern, rather, it seems more appropriate to determine basic word order dependent on the particular verb. Apart from that, Kurz (2000) observes that definite NPs precede indefinite ones more often.

Müller (1999) accounts for the word order variation in German within Optimality Theory (Prince & Smolensky, 2004). Similar to Uszkoreit (1987), he formulates a number of competing constraints which predict the most appropriate word order. The factors he considers concern grammatical form, information structure, animacy, etc.

Keller (2000) deals with the gradience in grammar and investigates diverse linguistic phenomena which require a gradient analysis as opposed to the dichotomy 'grammatical vs. ungrammatical'. In a series of psycholinguistic experiments with native speakers, he discovers hard and soft constraints and demonstrates that hard constraints are not context-dependent. With respect to German word order, Keller (2000) analyzes the effect case, pronominalization, verb position and information structure might have on it and formulates ranked constraints in the spirit of Optimality Theory based on the results of the experiments.

[4] www.coli.uni-saarland.de/projects/sfb378/negra-corpus
[5] http://corp.hum.ou.dk/itwebsite/corpora/corp/page18.html

Kempen & Harbusch (2004c) look at German subordinate clauses with intransitive, monotransitive and ditransitive verbs and consider several possibilities for the NP to be realized: either as pronouns or as full NPs. Comparing subjects (+SUBJ), direct (+DOBJ) and indirect objects (+IOBJ), they found that if pronominalized their order is invariably

$$[+\text{SUBJ}+\text{PRO}] \prec [+\text{DOBJ}+\text{PRO}] \prec [+\text{IOBJ}+\text{PRO}]$$

Another finding is that the subject as a full NP may precede pronouns but not intervene between them. The predominant order for full NPs may vary but the predominant order is

$$[+\text{SUBJ-PRO}] \prec [+\text{DOBJ-PRO}] \prec [+\text{IOBJ-PRO}]$$

The inverted order [+IOBJ-PRO] $\prec$ [+SUBJ-PRO] is restricted to clauses with intransitive verbs, and [+DOBJ-PRO] $\prec$ [+IOBJ-PRO] only occurs as standard order licensed by special ditransitive verbs. The conclusion the authors draw from their corpus analysis is that linearization constraints are more complex and less flexible than is standardly assumed. These constraints do not involve a single feature but a feature combination. The constraints suggest a linearization system where individual constituents receive absolute rather than relative positions.

Kempen & Harbusch (2004a) pose the question of what factors control the actual linearization preferences. They point to different approaches, utilizing such linguistic features as syntactic role, pronominalization, definiteness, animacy, (but not givenness and length) and concentrate on animacy. Animacy can affect the order either indirectly – animate entities are usually actors, and these are usually subjects, or directly – animate NPs should precede what is inanimate. The results obtained from a corpus analysis strongly confirm the direct influence hypothesis, but it is definitely only one of the factors as, e.g., pronouns still precede animate NPs. Based on this finding, the authors speculate that the animacy should have double effect on sentence generation: first, during the "functional" stage when the syntactic roles are assigned, and then during the "positional" stage.

Kempen & Harbusch (2004b) suggest a corpus-based generation method which generates all possible orders of the arguments of a head verb in a subordinate clause. They take the following "determinants" into consideration: grammatical function, form, NP length, animacy and definiteness. They point out that lexical properties of the head verb (e.g., transitivity type, reflexivity, thematic relation) do matter. They observe the tendency for animate arguments to precede inanimate ones which can be attributed neither to definiteness of the NP (because animacy and definiteness are not correlated) nor to the length. They also observe that animate NPs are on average shorter.

Weber & Müller (2004) investigate the NEGRA corpus to find the parameter which would explain the [+SUBJ] $\prec$ [+OBJ] ordering in the German main clause the best way. The results indicate that "given" subjects tend to precede new objects, definite subjects precede indefinite

objects, and pronominalized subjects precede full NP objects. None of the preferences is absolute because reverse orderings can be found in each case. None of the three basic order patterns for givenness, definiteness, and pronominalization is confirmed for the *obj-subj-verb* order. With givenness and pronominalization, both orders occurred about equally often, and with definiteness the preference was reverse. The authors interpret the results in that subjects are more likely to be definite, regardless of word order and observe the following:

> "Definiteness did not interact with information structure as we would have expected" (Weber & Müller, 2004, p. 76).

Their conclusion is that linearization principles are soft constraints, and only their combination matters.

Pappert et al. (2007) present another study where the order among the verb arguments in German is discussed and such parameters as animacy, definiteness, case and dative constraint are investigated in a series of corpus queries, completion questionnaires, and self-paced reading experiments. The scope of the constraints on constituent order is restricted to the relative order of the two objects in double object sentences with the subject in the VF. The study reveals that dative objects tend to precede accusative ones and that animacy may affect the ordering, but that the validity of the definiteness constraint is questionable:

> "The association of Case, Animacy, and Definiteness raises the question of a common underlying factor" (Pappert et al., 2007, p. 326).

Following Lamers & de Hoop (2005), the authors suggest that there is one underlying constraint referring to Prominence that governs the realization of objects. Prominent constituents are, for instance, definite dative constituents with animate referents. The authors admit that the scope of the study is highly restricted and that the conclusion should not be overgeneralized.

With respect to information structure, our experiments demonstrated (see Chapter 5) that in German the VF tends to be occupied either by frame-setting topics, or non-established addressation topics. The former has also been claimed by Speyer (2005). Established topics, on the other hand, tend to appear in the beginning of the MF (Frey, 2004). Since these are usually referred to with a pronoun, this complicates the constraint formulated in earlier studies which states that pronouns tend to precede full NPs.

6.4.2 Motivation for a Machine Learning Approach

As we have seen, many factors influence the constituent order, and their interaction can be quite complex in some cases. Apart from the constraints posed by the grammar, information structure, surface form, animacy and discourse status have also been shown to be relevant.

Below we present a summary of the relevant factors which further motivate our choice of features for supervised machine learning of ordering preferences:

- constituents in the nominative case precede those in other cases, and dative constituents often precede those in the accusative case;
- the order of verb arguments depends on the subcategorization properties of the verb;
- constituents with a definite article precede those with an indefinite one;
- pronominalized constituents precede non-pronominalized ones;
- animate referents precede inanimate ones;
- short constituents precede longer ones;
- the preferred topic position is right after the verb;
- the initial position is usually occupied by scene-setting elements and topics.
- there is a default order based on semantic properties of constituents:

$$\text{Actor} \prec \text{Temporal} \prec \text{SpaceLocative} \prec \text{Means} \prec \text{Addressee} \prec \text{Patient} \prec \text{Source} \prec \text{Destination} \prec \text{Purpose}$$

The fact that a single scale or a combination of two scales is not enough to account for the variety of grammatical orders motivates an approach which takes a cumulation of all the factors listed above to determine the position of a constituent in a clause. It seems implausible that a set of rules can be defined within Optimality Theory, or any other linguistic framework, which would take into account all these factors and still be comprehensible and easily interpretable. A statistical approach is more appealing as it allows to encode many parameters as features and learn complex interactions which are no longer visible even to the eye of a very skilled linguist.

An important premise, which our method of constituent order generation is based on, is that generally all the linguistic factors we mentioned contribute to what we are going to call the **prominence** or **weight** of a constituent. And it is this prominence which determines the relative order of two constituents in the MF. The VF, as we have seen in the previous chapter, plays an important role in ensuring local coherence and making transitions between adjacent sentences smooth. Therefore, the reasons of why a constituent is placed in the VF go beyond prominence in our sense, as what could be more prominent than the pronominalized animate subject whose preferred position is still in the beginning of the MF as we have shown. The simple ordering rule is then as follows: in the MF, more prominent constituents precede less

prominent ones. Such properties as animacy, accessibility (manifested in the surface form as pronoun or definite NP), higher syntactic or semantic role (subject or actor) make a constituent more prominent and ensure its position closer to the beginning of the MF. Unlike them, long full inanimate NPs are considerably less prominent and thus appear to the end of the sentence. These two examples describe the extreme cases whereas the prominence of most constituents lies somewhere between them. From this perspective, the order of constituents in the MF can be determined straightforwardly once one has a means of determining the relative prominence for a pair of constituents. A more prominent one should always precede the less prominent one. This can be see as an extension of the "heavy NP shift" proposal in that there is also a linear precedence defined over a set of constituents, but in our case there are many linguistic factors contributing to the "heaviness" and not just the length in words.

The above considerations motivate our TWO-STEP method of constituent ordering which first fills the VF, and then orders the remaining constituents in the MF by sorting them. The details of the implementation of TWO-STEP follow in the end of the next section.

6.4.3 Implemented Methods

6.4.3.1 The RANDOM Baseline

The bottom line in the evaluation experiments is provided by the baseline which puts the constituents in a random order without taking any linguistic information into account.

6.4.3.2 The RAND_IMP Baseline

We improve the trivial random baseline described above by the three syntax-oriented rules which are motivated by the fact that German is a SVO/SOV[6] (or only SOV in Generative Grammar (Ouhalla, 1994)) language where the subject by default precedes the object:

1. The VF is reserved for the subject.
2. The second position (i.e., right after the verb in the main clause) is reserved for the direct object if there is any.
3. The order of the remaining constituents is generated randomly.

6.4.3.3 The SYNT-SEM Baseline

In searching for the correct order, similar to Ringger et al. (2004), we select the order with the highest probability conditioned on syntactic and semantic categories. Unlike them, we use

[6] S, V and O stand for subject, verb and object respectively.

dependency parses and compute the probability of the top clause node only, which is modified by all constituents. With these adjustments the probability of an order O given the history h, if conditioned on syntactic functions of constituents ($s_1 ... s_n$), is simply:

$$P(O|h) = \prod_{i=1}^{n} P(s_i|s_{i-1}, h) \tag{6.11}$$

Ringger et al. (2004) do not make explicit, what their set of semantic relations consists of. From the example in the paper, it seems that these are a mixture of lexical and syntactic information[7]. Our annotation does not specify semantic relations between constituents. Instead, some of the constituents are categorized as *pers, loc, temp, org* or *undef_ne* if their heads bear one of these labels (see Sec. 2.1.2). By joining these with possible syntactic functions, we obtain a larger set of syntactic-semantic tags as, e.g., *subj-pers, pp-loc, adv-temp*. We transform each clause in the training set into a sequence of such tags, plus three tags for the verb position (*v*), the beginning (*b*) and the end (*e*) of the clause. Then we compute the bigram probabilities[8].

Thus, our third baseline (SYNT-SEM) selects from all possible orders the one with the highest probability as calculated with the following formula:

$$P(O|h) = \prod_{i=1}^{n} P(t_i|t_{i-1}, h) \tag{6.12}$$

where t_i is from the set of joined tags. For Example (6.5), possible tag sequences (i.e., orders) are *'b subj-pers v pred pp e', 'b pp v subj-pers pred e', 'b pred v subj-pers pp e'*, etc. Note, that given that the longest clause has ten constituents, the algorithm requires up to 10! permutations for every clause in order to find the one with the highest probability.

6.4.3.4 The UCHIMOTO Baseline

As the fourth baseline we train a maximum entropy learner (OpenNLP[9]) and reimplement the algorithm of Uchimoto et al. (2000) (see Sec. 6.2 for the description). For every possible permutation, its probability is estimated according to the following formula (copied from

[7] For example *DefDet, Coords, Possr, werden*

[8] We use the CMU Toolkit (Clarkson & Rosenfeld, 1997) to compute the probabilities.

[9] http://opennlp.sourceforge.net

Eq. (6.2)):

$$\begin{aligned} P(1|h) &= P(\{W_{i,i+j} = 1 | 1 \leq i \leq n-1, 1 \leq j \leq n-i\}|h) \\ &\approx \prod_{i=1}^{n-1} \prod_{j=1}^{n-i} P(W_{i,i+j} = 1 | h_{i,i+j}) \\ &= \prod_{i=1}^{n-1} \prod_{j=1}^{n-i} P_{ME}(1|h_{i,i+j}) \end{aligned} \tag{6.13}$$

The task of the binary classifier is to predict the probability that the order of a pair of constituents is correct – $P_{ME}(1|h_{i,i+j})$. Figure 6.4 illustrates the training and the testing phases. The prediction is made based on the following features describing the verb or h_c – the head of a constituent c[10]:

vlex the lemma of the root of the clause (non-auxiliary verb);

vpass the voice of the verb;

vmod the number of constituents to order;

lex the lemma of h_c or, if h_c is a functional word, the lemma of the word which depends on it (e.g., for PPs the noun is taken, because it is the preposition which is dependent on the verb);

pos part-of-speech tag of h_c;

sem if defined, the semantic class of *c*; e.g., *im April 1900* and *mit Albert Einstein (with Albert Einstein)* are classified *temp* and *pers* respectively;

syn the syntactic function of h_c;

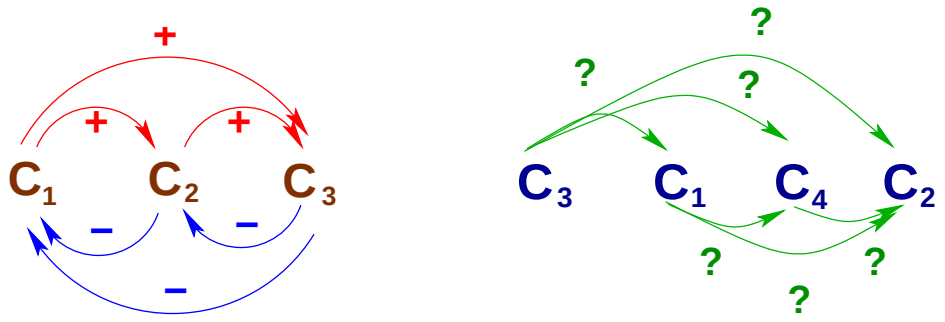
same whether the syntactic function of the two constituents is the same;

mod number of modifiers of h_c;

rep whether h_c appears in the preceding sentence;

pro whether *c* contains a (anaphoric) pronoun.

[10] We exclude features which use information specific to Japanese and non-applicable to German (e.g., on postpositional particles).



(a) Generating training instances from the data

(b) Estimating the probability of the order

Figure 6.4: The training and testing phases of the UCHIMOTO baseline.

6.4.3.5 The MAXENT Method

The first configuration of our system is an extended version of the UCHIMOTO baseline (MAXENT). To the features describing *c* we added the following ones:

det the kind of determiner modifying h_c (*def, indef, non-appl*);

rel whether h_c is modified by a relative clause (*yes, no, non-appl*);

dep the depth of *c*;

len the length of *c* in words.

The first two features describe the discourse status of a constituent; the other two provide information about its "weight". Since our learner treats all values as nominal, we discretized the values of **dep** and **len** with a C4.5 classifier (Kohavi & Sahami, 1996).

6.4.3.6 The TWO-STEP Method

The main difference between our first algorithm MAXENT and this one (TWO-STEP) is that we generate the order in two steps:

1. For the VF, using the OpenNLP maximum entropy learner for a binary classification (VF vs. MF), we select the constituent c with the highest probability of being in the VF, i.e., $\arg\max_c P(c|VF)$. Figure 6.5a illustrates the process with an example where from a set of four constituents (c_1, c_2, c_3, c_4) one is selected (c_3).

2. For the MF, the remaining constituents are put into a random order and then sorted. The training data for the second task is generated only from the MF of clauses. Figure 6.5b gives an example where the three constituents in the VF are sorted with only three comparisons.

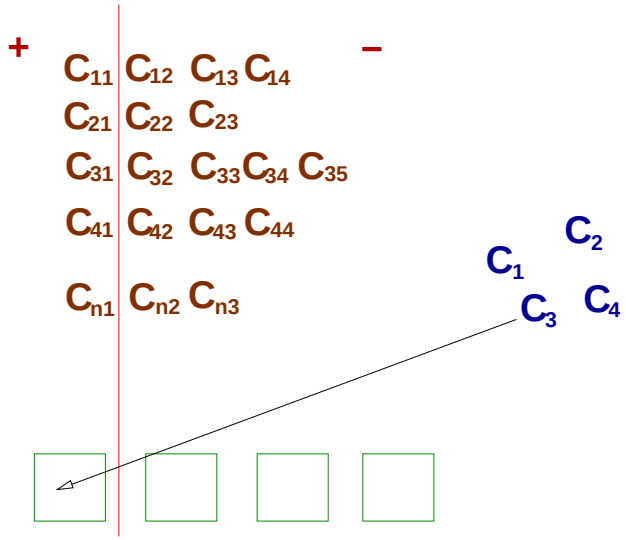
Another modification concerns the efficiency of the algorithm. Instead of calculating probabilities for all pairs, we obtain the right order from a random one by **sorting**. We compare adjacent elements by consulting the learner as if we would sort an array of numbers with Bubble Sort (the efficiency of the sorting algorithm is not important here). Given two adjacent constituents, c_i and c_j such that c_i precedes c_j, we compute the probability of them being in the right order, i.e., $P(c_i \prec c_j)$. If it is less than 0.5, we transpose the two and compare c_i with the next adjacent constituent.

Since the sorting method presupposes that the predicted relation is transitive, we checked whether this is really so on the development and test data sets. We looked for three constituents c_i, c_j, c_k from a sentence S, such that $P(c_i \prec c_j) > 0.5$, $P(c_j \prec c_k) > 0.5$, $P(c_i \prec c_k) < 0.5$ and found none. Therefore, unlike UCHIMOTO, where one needs to make exactly $N! \times N(N-1)/2$ comparisons select the best order of N constituents, we need to make $N(N-1)/2$ comparisons at most. Figure 6.6 presents an implementation of ORDER-CONSTITUENTS(*cons*) from the algorithm in Figure 6.3 with TWO-STEP. Functions SELECT-FOR-VF(*cons*) removes the best candidate for the VF from the given set of constituents (*cons*) and returns it back. Functions RANDOMIZE-ORDER(*cons*) and SORT(*cons*) are void and do what they stand for: the former brings the constituents in a list into a random order; the latter sorts constituents by estimating the probability $P(c_i \prec c_j)$. The choice of the sorting algorithm is not important for us as the number of constituents to order hardly ever exceeds eight.

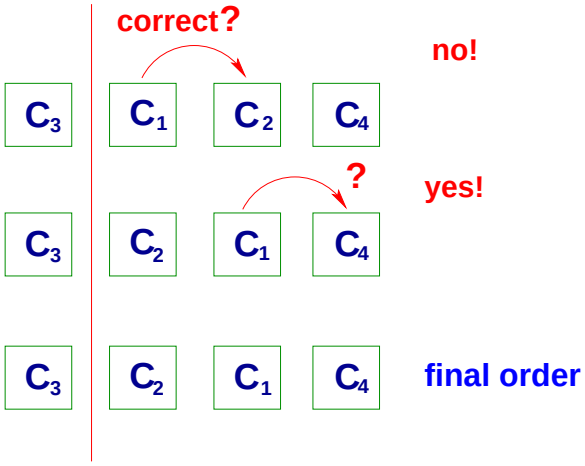
In the implementation of our fusion system, to make the treatment of main and hypotactic clauses similar, we use the first classifier also to find the best candidate for the first position in the subordinate clauses. There, the classifier learns to assign high probability to subordinate conjunctions (e.g., *dass (that), weil (because), obwohl (although)* etc.) as well as to relative pronouns (*die (which/that/who), dessen (whose)* etc.). Although this could be done with a few simple rules, we prefer to train an extra classifier in order not to treat hypotactic clauses as a special case.

6.4.4 Experiments

The goal of our experiments is to check whether our TWO-STEP method outperforms other baselines, in particular the one which utilizes the same set of features but does not differentiate between the VF and the MF. The first 100 files from WikiBiography (Sec. 2.1.2) are used for testing. All classifiers are trained on about 900 files (from 400 to 1,267).



(a) Selecting the best candidate for the VF



(b) Ordering constituents in the MF by sorting

Figure 6.5: Two-Step method of ordering constituents

function ORDER-CONSTITUENTS($cons$) **returns** List

1: Constituent $v \leftarrow$ SELECT-FOR-VF($cons$)
2: RANDOMIZE-ORDER($cons$)
3: SORT($cons$)
4: $cons$.`insert`$(0, v)$
5: **return** $cons$

function SELECT-FOR-VF($cons$) **returns** Constituent

1: $max \leftarrow -1, best \leftarrow$ **null**
2: **for all** $c \in cons$ **do**
3: **if** $P(VF|c) > max$ **then**
4: $max \leftarrow P(VF|c), best \leftarrow c$
5: **end if**
6: **end for**
7: $cons$.`remove`$(best)$
8: **return** $best$

Figure 6.6: Implementation of ORDER-CONSTITUENTS(*cons*) with TWO-STEP

6.4.4.1 Evaluation Metrics

We use four metrics to automatically evaluate our systems and the baselines. Of course, evaluation with native speakers, who can reliably distinguish between appropriate, acceptable, grammatical and ungrammatical orders, would be better than automatic evaluation. The results of an evaluation with human judges is presented in Chapter 7. The metrics are presented below:

acc The first is per-clause **accuracy** which is simply the proportion of correctly regenerated clauses:

$$acc = \frac{|correct|}{|total|} \tag{6.14}$$

Clearly, this metric evaluates the performance rigorously and gives a zero score to any order different from the source one which in some cases is too harsh (see Cahill & Forst (2009)).

τ Kendall's τ, which has been used for evaluating sentence ordering tasks (Kendall, 1938; Lapata, 2006), is the second metric we use:

$$\tau = 1 - 4\frac{t}{N(N-1)} \tag{6.15}$$

where t is the number of interchanges of consecutive elements to arrange N elements in the right order. τ is sensitive to near misses and assigns *abdc* (almost correct order) a score of 0.66 while *dcba* (inverse order) gets -1. Note that it is questionable whether this metric is as appropriate for word ordering tasks as for sentence ordering ones. Sentences in (6.4.4.1) provide an example where, given (6.16) as source sentence, a near miss (6.17) is ungrammatical whereas an order which requires more swaps (6.18) is acceptable (the values of τ are 0.67 and 0, respectively).

(6.16) [After the lesson] [the teacher] sent [them] [to the principle].

(6.17) *[The teacher] sent [after the lesson] [them] [to the principle].

(6.18) [The teacher] sent [them] [to the principle] [after the lesson].

agr Another metric we use is **agreement rate** which was introduced by Uchimoto et al. (2000):

$$agr = \frac{2p}{N(N-1)} \tag{6.19}$$

i.e., the number of correctly ordered pairs of constituents over the total number of all possible pairs. Uchimoto et al. (2000) in their experiments also use **complete agreement** which is basically per-clause accuracy. Unlike τ, which has -1 as the lowest score, *agr* ranges from 0 to 1.

inv Ringger et al. (2004) evaluate the performance only in terms of **per-constituent edit distance**, *ped*, calculated as follows:

$$ped = \frac{m}{N} \tag{6.20}$$

where m is the minimum number of 'moves' – a move is a deletion combined with an insertion – needed to put N constituents in the right order. This measure is different from τ or *agr* in that it does not take the distance of the move into account and scores *abced* and *eabcd* equally (0.2). Since τ and *agr*, unlike *edit distance*, give higher scores to better orders, we compute **inverse distance** instead:

$$inv = 1 - ped \tag{6.21}$$

Thus, all three metrics (τ, *agr, inv*) give the maximum of 1 if constituents are ordered correctly. Just as τ, *agr* and *inv* can give a positive score to an ungrammatical order. Hence, none of the evaluation metrics describes the performance perfectly.

	acc	τ	*agr*	*inv*
RANDOM	15%	0.02	0.51	0.64
RAND_IMP	23%	0.24	0.62	0.71
SYNT-SEM	51%	0.60	0.80	0.83
UCHIMOTO	50%	0.65	0.82	0.83
MAXENT	52%	0.67	0.84	0.84
TWO-STEP	61%	0.72	0.86	0.87

Table 6.1: Per-clause mean of the results

	acc	τ	*agr*	*inv*
TWO-STEP VF	68%	-	-	-
TWO-STEP MF	80%	0.92	0.96	0.95

Table 6.2: Mean of the results for the VF and the MF (main clauses)

6.4.4.2 Results

The results on the test data are presented in Table 6.1. The performance of TWO-STEP is significantly better than that of any other method (χ^2, $p < 0.01$). The performance of MAXENT does not significantly differ from UCHIMOTO. SYNT-SEM performed about as good as UCHIMOTO and MAXENT. We also checked how well TWO-STEP performs on each of the two sub-tasks (Table 6.2) and found that the VF selection is considerably more difficult than the sorting part.

The most important conclusion we draw from the results is that the gain of 9% accuracy is due to the VF selection only, because the feature sets are identical for MAXENT and TWO-STEP. From this follows that doing feature selection without splitting the task in two is ineffective, because the importance of a feature depends on whether the VF or the MF is considered. For the MF, feature selection has shown **syn** and **pos** to be the most relevant features. They alone bring the performance in the MF up to 75%. In contrast, these two features explain only 56% of the cases in the VF. This implies that the order in the MF mainly depends on grammatical features, while for the VF all features are important because removal of any feature caused a loss in accuracy.

Another important finding is that there is no need to overgenerate to find the right order. Insignificant for clauses with two or three constituents, for clauses with 10 constituents, the number of comparisons is reduced drastically from 163,296,000 to 45.

According to the *inv* metric, our results are considerably worse than those reported by

Ringger et al. (2004). However, the fact that they generate the order for every non-terminal node seriously inflates their numbers. Apart from that, they do not report accuracy, and it is unknown, how many sentences they actually reproduced correctly. Another reason might be related to the differences in the data used. Ringger et al. (2004) perform experiments on a corpus of computer manuals which seem to contain shorter and simpler sentences than biographies do.

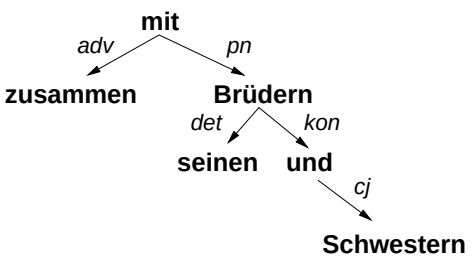
6.4.4.3 Error Analysis

To reveal main error sources, we analyzed incorrect predictions concerning the VF and the MF, one hundred for each. Most errors in the VF did not lead to unacceptability or ungrammaticality. From lexical and semantic features, the classifier learned that some expressions are often used in the beginning of a sentence. These are temporal or locational PPs, anaphoric adverbials, some connectives or phrases starting with *unlike X*, *together with X*, *as X*, etc. Such elements were placed in the VF instead of the subject found in the VF in the source sentences and this caused an error although both variants were equally acceptable. In other cases the classifier could not find a better candidate but the subject because it could not conclude from the provided features that another constituent would nicely introduce the sentence into the discourse. Mainly this concerns recognizing information familiar to the reader not by an already mentioned entity, but one which is inferrable from the preceding context.

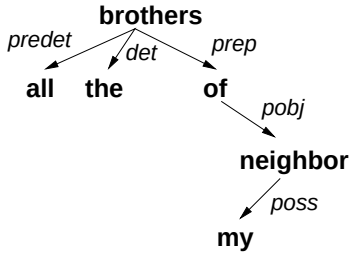
In the MF, many orders had a PP transposed with the direct object. In some cases the predicted order seemed as good as the correct one. Often, the algorithm failed at identifying verb-specific preferences: For example, some verbs take PPs with the locational meaning as an argument and normally have them right next to them, whereas others do not. Another frequent error was the wrong placement of superficially identical constituents, e.g., two PPs of the same size. To handle this error, the system needs more specific semantic or subcategorization information. Some errors were caused by the parser, which created extra constituents (e.g., wrong PP or adverb attachment) or confused the subject with the direct object.

6.4.4.4 Summary and Discussion

In this section we presented six constituent ordering methods: two trivial baselines, a baseline which relies on syntactic and semantic information, a baseline which utilizes a richer linguistic representation and two of our methods. The non-trivial baselines achieve an accuracy of 50% which is significantly higher than that of the random baselines. Additional features which aim at modeling the discourse status and the "heaviness" of constituents bring a slight improvement whose significance is inconclusive. The significantly better performance of TWO-STEP supports our hypotheses articulated in Chapter 5 and in Section 6.4.2 concerning (i) the special status of the VF, as well as (ii) the prominence of constituents to which their linguistic



(a) Tree corresponding to the PP *zusammen mit seinen Brüdern und Schwestern*



(b) Tree corresponding to the NP *all the brothers of my neighbor*

Figure 6.7: Trees of a German PP and an English NP

properties from different levels of analysis contribute.

The accuracy of 61% is encouraging as it gives us evidence that the task of constituent ordering can be solved efficiently and reliably at least on our corpus. Further error analysis revealed that a considerable portion of errors did not result in ungrammatical sentences. Therefore, the amount of grammatical orders produced by our algorithm is even higher than 61%. Presumably, the results might be improved with a careful feature analysis and feature combinations. However, we did not do any experiments in this direction due to two reasons. On the practical side, we do not expect feature combinations to improve the performance dramatically, e.g., with another ten or five percent. On the theoretical side, the improvement of 10% as it is already provides a solid support for our hypotheses.

6.5 Linearizing Constituents

To linearize constituent subtrees, we use the 'overgenerate and rank' approach described in Section 6.2 and rely on a trigram LM to rank linearizations. In this section we present the results not only on German but also on English data and show that even a rigid word order language such as English benefits from a combined approach. In particular, we will argue the following:

1. That trigram LMs are well-suited for constituent linearization.

2. That there is a considerable drop in performance when one uses them on the clause level for constituent ordering.

3. That an approach which uses a richer representation on the clause level is more appropriate for German as well as for English.

6.5.1 Method Description

Given a projective dependency tree, all linearizations can be found recursively by generating permutations of a node and its children. Figure 6.8 presents the recursive algorithm for finding the linearization variants – GET-POSSIBLE-ORDERS(*node*). The algorithm first finds all possible linearizations for each child of a node, and then generates all admissible permutations of the children and the node itself. Unfortunately, the number of possible permutations grows factorially with the branching factor and exponentially with the depth of the tree. Hence, it is highly desirable to prohibit generation of clearly unacceptable permutations by putting hard constraints encoded in the German and English grammars. The constraints which we implement in our study are the following:

- Determiners (apart from the articles, these include possessives and quantifiers) and noun or adjective modifiers always **precede** their heads.
- Conjunctions, coordinated elements, prepositional objects and appositions (in German) always **follow** their heads.

These constraints allow us to limit, e.g., the total of 96 ($2\times2\times4!$) possibilities for the tree corresponding to the phrase *all the brothers of my neighbor* (see Fig. 6.7b) to only two (*all the brothers of my neighbor, the all brothers of my neighbor*). Similarly, for the German phrase *zusammen mit seinen Brüdern und Schwestern* (*together with his brothers and sisters*) (Fig. 6.7a), from the total of 72 ($3! \times 3! \times 2$) projective variants only two are valid – the original one and *mit seinen Brüdern und Schwestern zusammen*. The function IS-VALID-SEQUENCE(*kids, i, head*) checks whether an insertion of the head *node* in the *i*th position among its children results in an impossible sequence. For example, IS-VALID-SEQUENCE($\{\{$*all*$\}$, $\{$*the*$\}$, $\{$*of my neighbor*$\}\}$, *1, brothers*) returns **false** because the head *brothers* is not allowed to precede the article *the*.

Apart from the constraints from the grammar, in the sentence fusion setting one may assume that if two words in the fused tree are found in the same order in all the input sentences, then their order is unlikely to change for the output. For example, given that *alte* (*old*) precedes *große* (*big*) in every input sentence, one does not need to consider linearization variants where the order is reverse, i.e., *große alte*. (The function IS-VALID-SEQUENCE(*kids, i, head*) does not implement these constraints; see the software package for details.) Even with the constraints described above, in some cases the list of possible linearizations is too long and has to be reduced to the first N, where N is supposed to be sufficiently large. In our experiments we break the permutation generation process if the limit of 20,000 variants is reached (see Line 10 in the GET-PERMUTATIONS-WITH-HEAD(*kids, head*) function, Fig. 6.8). However, this happens very rarely as we will demonstrate in the results section.

```
function GET-POSSIBLE-ORDERS(node) returns List
    List kids ← {}
    for all c ∈ node.children() do
      kids.add(GET-POSSIBLE-ORDERS(c))
    end for
    return GET-PERMUTATIONS-WITH-HEAD(kids, node)

function GET-PERMUTATIONS-WITH-HEAD(kids, head) returns List
 1: if kids.size() = 0 then
 2:   return {head}
 3: end if
 4: List perms ← GET-PERMUTATIONS(kids), orders ← {}
 5: for all p ∈ perms do
 6:   for all 0 < i ≤ kids.size() do
 7:     if IS-VALID-SEQUENCE(p, i, head) then
 8:       List order ← INSERT(p, i, {head})
 9:       orders.addAll(GET-ALL-COMBINATIONS(order))
10:       if orders.size() > 20,000 then
11:         return null
12:       end if
13:     end if
14:   end for
15: end for
16: return orders

function IS-VALID-SEQUENCE(kids, i, head) returns boolean
    for all 0 < j < i do
      if MUST-FOLLOW(kids.get(j), head) then
        return false
      end if
    end for
    for all i ≤ j < kids.size() do
      if MUST-PRECEDE(kids.get(j), head) then
        return false
      end if
    end for
    return true
```

Figure 6.8: The recursive algorithm GET-POSSIBLE-ORDERS(*node*)

```
function GET-ALL-COMBINATIONS(order) returns List
    List l ← order.get(0)
    for all 1 < i < order.size() do
      l ← COMBINE(l, order.get(i))
    end for
    return l

function COMBINE(a, b) returns List
    List l ← {}
    for all i ∈ a do
      for all j ∈ b do
        l.add(CONCATENATE(i, j))
      end for
    end for
    return l
```

Figure 6.9: Methods for getting all variants for a sequence of children and their head

We do not present the pseudo-code for GET-PERMUTATIONS(*kids*) (Line 1 in Fig. 6.8) which simply returns a list of *n!* orders for *n* sibling nodes. In fact, it returns a list of lists of lists because a permutation is a list itself where each node is a list of its possible linearizations. We do not detail the CONCATENATE function which simply concatenates two given strings. The INSERT function takes a list of lists corresponding to children nodes and inserts a given list of a single head node to a specified position. As an illustration consider the NP *a very complex function*, where *function* is the head modified by *a* and *complex*, the latter being modified by *very*.

1. GET-POSSIBLE-ORDERS(*a*) **returns** {*a*}
2. GET-POSSIBLE-ORDERS(*very*) **returns** {*very*}.
3. GET-PERMUTATIONS({{*very*}}) **returns** {{{*very*}}}.
4. INSERT({{*very*}},*0*, {*complex*}) **returns** {{*complex*}, {*very*}}.
5. GET-ALL-COMBINATIONS({{*complex*}, {*very*}}) **returns** {*complex very*}.
6. GET-PERMUTATIONS-WITH-HEAD({{*very*}}, *complex*) **returns** {*complex very*, *very complex*}.

7. GET-PERMUTATIONS($\{\{a\}, \{$*complex very, very complex*$\}\}$) **returns** $\{\{\{a\}, \{$*complex very, very complex*$\}\}, \{\{$*complex very, very complex*$\}, \{a\}\}\}$.

8. IS-VALID-SEQUENCE($\{\{a\}, \{$*complex very, very complex*$\}\}$, *i*, *function*) **returns false** for $i = 0$.

9. IS-VALID-SEQUENCE($\{\{$*complex very, very complex*$\}, \{a\}\}$, *i*, *function*) **returns false** for $i < 2$.

10. GET-PERMUTATIONS-WITH-HEAD($\{\{a\}, \{$*complex very, very complex*$\}\}$) **returns** $\{$*a function complex very, a function very complex, a complex very function, a very complex function, complex very a function, very complex a function*$\}$.

Thus, the language model has to rank the total of six word sequences, only one of which is a grammatical English phrase. Without the grammar constraints, the number of possible sequences would be twice as big. Without the projectivity constraint, the number of possibilities would amount to 24.

6.5.2 Experiments

The purpose of our experiments is two-fold: first, we want to evaluate the usefulness of a trigram LM-based method in linearizing constituents. Second, we want to check whether the performance of the method is significantly worse for constituent ordering. Trigram LMs were built for German and English as follows:

German: We took the German Wikipedia dump from September 2007 which included all the articles but not the edit history. From those articles we excluded 100 biographies which we used as test data in the constituent ordering experiments – these constitute the test set for the present experiments too. Note that the annotated biographies we used for testing come from an earlier version of Wikipedia. However, since some sentences might have survived a year and a half of editions, we decided to exclude those from the data and generated a trigram LM from approx. 1.2G of text. All words but nouns (the *nn* and *ne* pos-tags) were lowercased.

English: We reserved a small subset of the annotated WSJ data for testing (about 600 articles, 340,000 words) and used the remaining part of the corpus (the WSJ articles from the period 1987-1992) to build a trigram LM. All data are lowercased.

Both trigram models were built with the CMU toolkit (Clarkson & Rosenfeld, 1997). We use Good-Turing smoothing and trained the models with the vocabulary size of about 52,000 words for both languages. The sentence boundaries were annotated automatically, and

a dummy sentence boundary tag was inserted in the beginning and the end of each sentence. All numbers were replaced with the *CARD* tag and punctuation was removed. Accordingly, in test instances the numbers were replaced with *CARD* too. Presumably, a model with annotated phrase boundaries (e.g., with a chunker) would be even more suitable but we have not experimented with that.

To test the trigram-based method on the clause level, we generate all possible permutations of clause constituents, insert the verb and then rank the resulting strings with the LM taking the information on sentence boundaries into account. The verb is inserted right after the subject in English clauses; in German clauses the inflected verb is placed after the first constituent in main clauses and in the end otherwise. The non-finite part of the verb is placed in the end of the clause.

We use TWO-STEP as a point of comparison for the trigram LM-based method on the clause level. The implementation of TWO-STEP for English is the same as for German. As semantic information is not included in the annotation of our English corpora, the **sem** feature does not apply. To train the maximum entropy classifiers needed by TWO-STEP we used about 41,000 sentences from the WSJ corpus (Sec. 2.2.2).

6.5.2.1 Evaluation Metrics

Although both languages allow for some variation in word order and it might happen that the generated order is not necessarily wrong if different from the original one, we do not expect this to happen often and evaluate the performance rigorously: only the original order counts as the correct one. Moreover, given the projectivity constraint it is highly unlikely that a modified order **within** constituents would result in an equally acceptable phrase. Therefore, the default evaluation metric for constituent linearization is per-constituent accuracy:

$$acc = \frac{|correct|}{|total|}$$

Other metrics we use to measure how different a generated order of N elements is from the correct one are:

1. Kendall's τ, $\tau = 1 - 4\frac{t}{N(N-1)}$ where t is the minimum number of interchanges of consecutive elements to achieve the right order.

2. Agreement rate, $agr = \frac{p}{N(N-1)}$ – the number of pairs in the correct order over the total number of possible pairs.

3. Edit distance related inv, $inv = 1 - \frac{m}{N}$ where m is the minimum number of deletions combined with insertions to get to the right order.

	acc	τ	*agr*	*inv*
non-triv	73%	0.84	0.92	0.93
> 1	85%	0.91	0.96	0.96
≥ 1	92%	0.95	0.97	0.98
overall	92%	0.95	0.97	0.98

(a) Constituent linearization results for German

	acc	τ	*agr*	*inv*
non-triv	76%	0.85	0.92	0.94
> 1	85%	0.90	0.95	0.96
≥ 1	91%	0.94	0.97	0.98
overall	90%	0.93	0.96	0.97

(b) Constituent linearization results for English

Table 6.3: Results of the trigram method for constituent linearization

We use these metrics to see how many transpositions are needed on average to bring wrong linearizations in the right order, i.e., to see how many of the wrong orders are in fact near misses.

6.5.2.2 Results

The results of the experiments on the constituent level in German and English are presented in Tables 6.3a and 6.3b respectively. In the first row of the tables in 6.3 (*non-triv*) we give the results for cases where, with all constraints applied, there were still several possible linearizations but where their number was less than 20,000 – these are precisely the cases where we consulted the LM to rank the orders. The second row (> 1) is for all constituents listed as *non-triv* plus all other constituents which were longer than one word. The third row (≥ 1) presents the results for all the constituents counted as > 1 plus single-word constituents. Finally, the bottom row gives the results for all the constituents, including cases with more than 20,000 linearizations (*overall*).

German: From the total of 5,000 constituents extracted from the dependency trees in the test set, only 3 (about 0.06%) were discarded because the number of admissible linearizations exceeded the limit of 20,000. Only for about one third of all constituents more than two linearizations were possible (1,574); almost half of all constituents were one word long (2,100). About quarter of all constituents (1,323) were longer than one word but their only admissible linearization could be found with the grammar constraints.

English: From the total of 5,000 constituents, about one half were one word long (2,155). The grammar constraints reduced the number of possibilities to one in every fifth case (994). For one third of the cases the LM was consulted (1,797) and in 1% of the cases the number of linearizations was too large (55).

To verify our earlier claim concerning the relative inappropriateness of trigram LM meth-

	acc	τ	*agr*	*inv*
TRIGRAM	47%	0.49	0.75	0.80
TWO-STEP	61%	0.72	0.86	0.87

(a) Constituent ordering results for German

	acc	τ	*agr*	*inv*
TRIGRAM	51%	0.45	0.72	0.81
TWO-STEP	68%	0.72	0.86	0.89

(b) Constituent ordering results for English

Table 6.4: Results of the two methods on the clause level

ods for constituent ordering, we also present the results of TRIGRAM for this task on German and English data and compare them against the results obtained with TWO-STEP (Table 6.4). Here, we discarded trivial cases and considered only clauses which had at least two constituents dependent on the verb. This filtering resulted in approx. 2,200 test clauses for English and for German each (the German results are copied from Table 6.1).

6.5.2.3 Summary and Discussion

The difference in accuracy between the performance of the trigram approach on the constituent and the clause level is considerable – 26% and 25% for German and English respectively (*non-triv* is compared with TRIGRAM). The high accuracy on the constituent level is remarkable given that the average length of constituents which counted as *non-triv* is about six words for both languages, whereas the average clause length in constituents is 3.3. The statistically significant difference in performance on the two tasks – constituent linearization and constituent ordering – supports our hypothesis that the trigram LM-based approach advocated in earlier studies is more adequate for finding the optimal order **within** constituents. This difference also demonstrates that, on the clause level, our TWO-STEP method which takes a range of grammatical features into account and splits the constituent ordering task in two is more appropriate.

The superior performance on the clause level on English data can be explained by the fact that English word (i.e., constituent) order is more rigid and is thus easier to model. The results on the constituent level are consistent between the two languages.

6.6 Summary

We presented a combined tree linearization algorithm which first linearizes constituents dependent on the verb with a trigram model and then orders clausal constituents taking a range of linguistic features into account. We demonstrated that the splitting of the linearization task into two is meaningful for two reasons. Firstly, trigram language models are reliable for finding the word order within constituents. Secondly, their performance drops significantly as soon as one applies them on the clausal level because trigrams are insufficient for recognizing

long distance dependencies. A method which takes several linguistic factors into account performs significantly better in constituent ordering. Moreover, we argued that the organization of the clause also requires a differentiated approach to constituent ordering. The TWO-STEP approach, which first selects the best starting point for the sentence and then orders the remaining constituents, by far outperforms several non-trivial baselines, including a trigram language model based one. Based on the research introduced in the previous chapter, we argued that in German, the task of determining the first constituent of the main clause is more difficult than the task of ordering the remaining constituents: the decision should be made by considering the factors from the morphological, syntactic, semantic and discourse levels.

The combined approach works equally well on English and German data. Interestingly, the accuracy of the trigram language model based method on the constituent and the clausal levels is comparable accross the two languages. Therefore, the freedom of word order in German concerns the level of clause constituents only. On the constituent level a simple trigram language model suffies to reliably predict the correct word order.

Chapter 7

Evaluating deFuser

In this chapter we describe an experiment assessing the performance of deFuser on the task of generic sentence fusion with respect to readability and informativity of fused sentences. deFuser is compared with two baselines. The first one is a random baseline which sets the upper and the lower bounds on readability and informativity respectively. The second one is a reimplementation of the fusion system of Barzilay & McKeown (2005) for German.

7.1 Goals of Evaluation

It is notoriously difficult to evaluate generation and summarization systems as there are many dimensions in which the quality of the output can be assessed. Given that the goal of text summarization is to produce a text which transmits the important content of the input, at least two important evaluation parameters should be distinguished. These correspond to the following two questions:

1. How good is the generated text from a linguistic point of view? This question can be further detailed with questions concerning the different facets of linguistic quality, such as grammaticality, meaningfulness, coherence.

2. Does the summary convey all and only important information from the source text(s)? This question can be complemented with the one about summary redundancy (i.e., whether the summary is repetitive).

Although automatic evaluation measures correlate with human judgments (Lin, 2004), they cannot fully replace them. Summarization competitions like DUC/TAC include both automatic and manual evaluation, and getting higher ranks in the latter is more important than scoring high in the automatic evaluation. An accurate and detailed linguistic analysis of an

utterance can be provided by professional linguists but cannot be expected from naïve speakers. Nonetheless, the latter are good at judging utterance acceptability. In our evaluation we ask the participants to assess sentence **readability** to find an answer to the first question posed above. Readability assumes both grammatical well-formedness and semantic soundness, so that grammatical but meaningless sentences get a low readability score. To find an answer to the second question, we ask human judges to assess **informativity** of fused sentences. Informativity stands for how well the fusion system selects and combines important content. The instructions for the human judges can be found in Appendix A.2.

Note that both parameters are crucial for a fusion system and that they are interdependent. Unreadable sentences have no utility simply because they fail at transmitting any sensible content and are very likely to score low on the informativity scale. Also, generating uninformative albeit grammatical sentences has little value for a practical application. In fact, a perfectly readable sentence can be "generated" by simply returning one of the input sentences.

7.2 Evaluation Design

In our evaluation we strive to assess deFuser with regard to readability and informativity by comparing its performance with two baselines. Being a complete text-to-text application, deFuser consists of several modules each of which may contribute to mistakes in the generated sentence. Figure 7.1 (copied from the introduction) gives an overview of the fusion pipeline from annotated documents to novel sentences. Apart from an overall evaluation of the system as a whole, it would be helpful to evaluate relative performance of each of the modules and identify "weaker links" in the pipeline. Recall that the main contribution of the present work concerns the sentence generation process which, in essence, consists of novel tree generation and tree linearization. Unlike these stages of the fusion pipeline, the sentence grouping procedure (see the sentence grouper box in Fig. 7.1) is fairly straightforward and draws upon previous algorithms to a large extent. To focus on the novel part of our system, we use the output of the sentence grouping module not only in deFuser but in other baselines, too. This way we can be sure that the systems are compared on an equal footing and that the poorer/better performance of deFuser is not due to the module we are least interested in.

We concluded Chapter 6 with an extensive evaluation and demonstrated that our linearization method – i.e., trigram LM for ordering words within phrases combined with TWO-STEP for constituent ordering (LM + TWO-STEP) – significantly outperforms several non-trivial baselines. That is, we already know how well our combined linearizer performs. Moreover, we have solid evidence that it works better than several other methods. What we know nothing about so far is how well the tree generation part of our system works. In order to evaluate exactly this part of deFuser (tree transformer combined with aligner/merger combined with

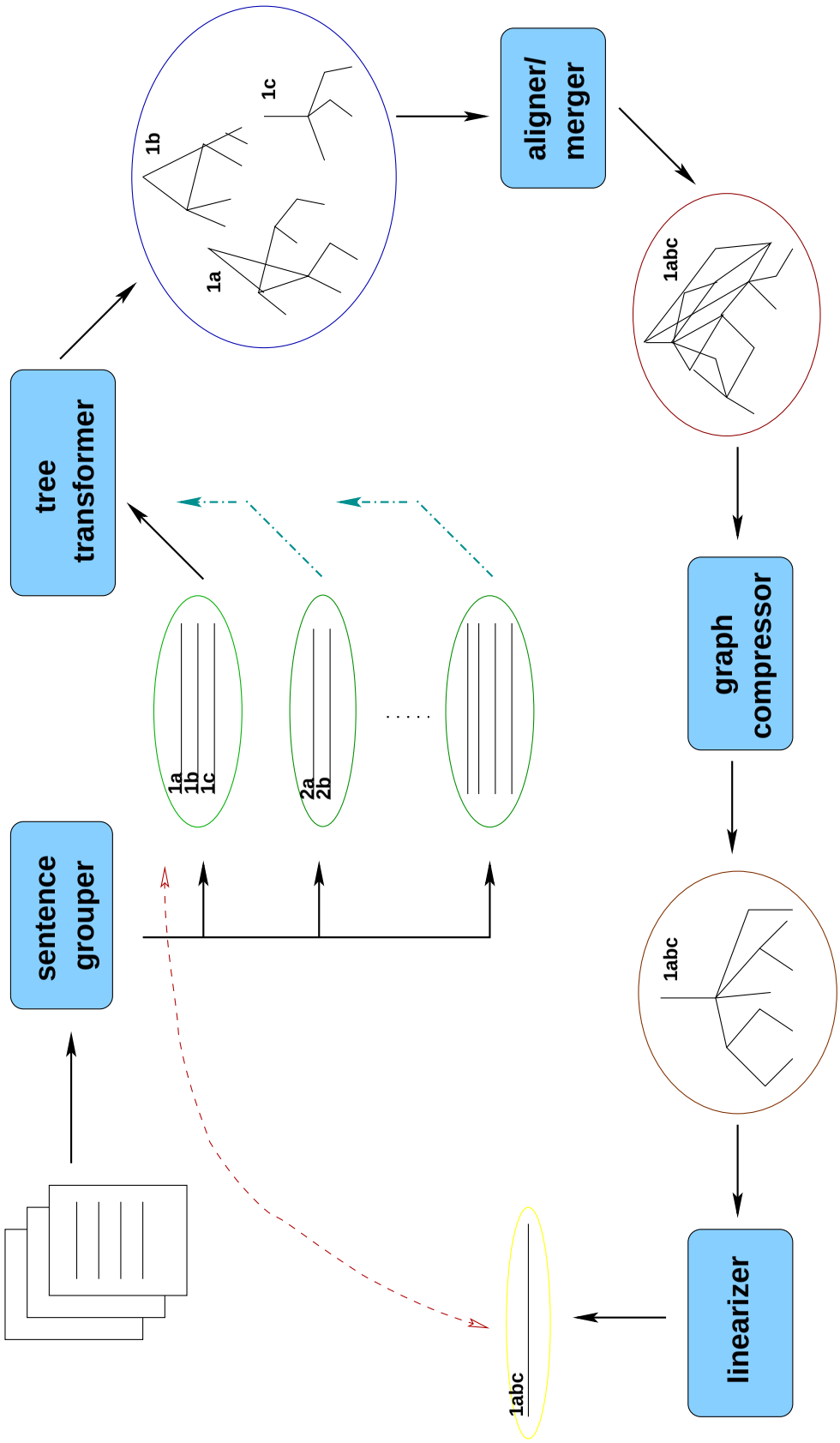


Figure 7.1: deFuser system overview

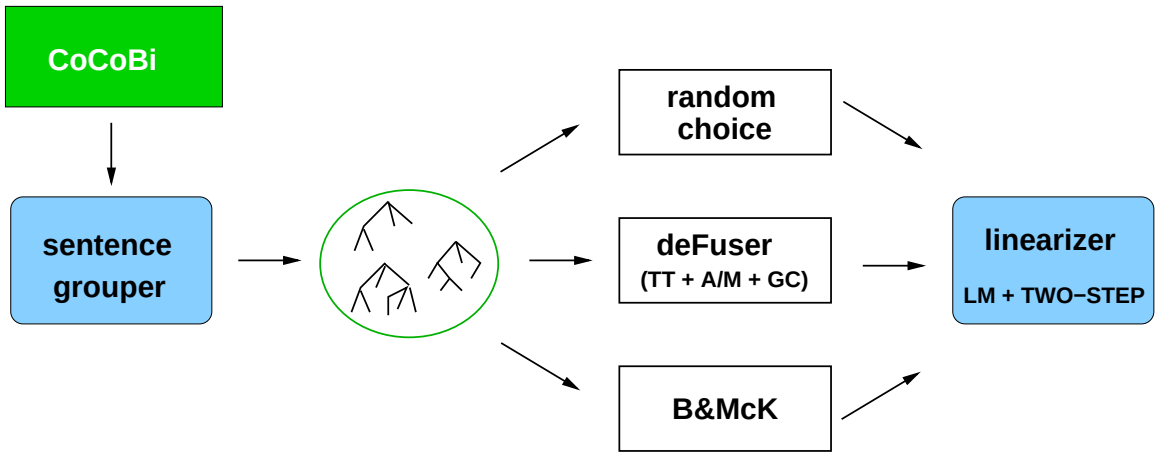


Figure 7.2: Fusion pipeline of deFuser and two baselines

graph compressor in Fig. 7.1), we utilize our combined linearization algorithm for all the systems. This way we exclude the possibility that poorer readability ratings of other systems are due to a less accurate linearization algorithm.

Figure 7.2 demonstrates the evaluation architecture. CoCoBi (Sec. 2.1.1) provides sets of annotated biographies of a person as input. Each of these sets is fed into the sentence grouper which in turn outputs groups of similar sentences. deFuser and the baselines take a group of such sentences as input and operate on their parse trees. Each of the three systems produces a dependency tree which is then sent to the linearizer implementing our combined tree linearization algorithm. deFuser builds new trees via transformation (TT), alignment/merging (A/M) and graph compression (GC) (all described in Chapter 4). In the rest of this section we present the baselines.

7.2.1 Random Baseline

The random baseline simply picks one of the input sentences and sends it to the linearizer. This trivial strategy may produce sentences of three kinds:

1. a sentence identical to the selected source sentence;
2. a grammatical sentence with an alternative ordering;
3. an ungrammatical sentence different from the source one.

We implemented this baseline because its performance sets an upper bound on readability and a lower bound on informativity. Indeed, assuming that the parse tree is correct, the baseline does not have a chance to make it ungrammatical because it sends the dependency tree directly

to the linearizer. The average rating of this baseline on the readability scale gives the actual boundary on how much we can expect given the parser we use and the combined linearizer. At the same time the random baseline sets the lower bound on informativity because it neither removes irrelevant information, nor combines important pieces of content from different sentences.

7.2.2 The Algorithm of Barzilay & McKeown, 2005

To date, the algorithm of Barzilay & McKeown (2005) is the first and the only example of an implemented and evaluated sentence fusion system. Therefore, we reimplemented and adapted it to process German data to have a point of comparison for deFuser. In this section we first describe the architecture of the original fusion system and then introduce our reimplementation for German.

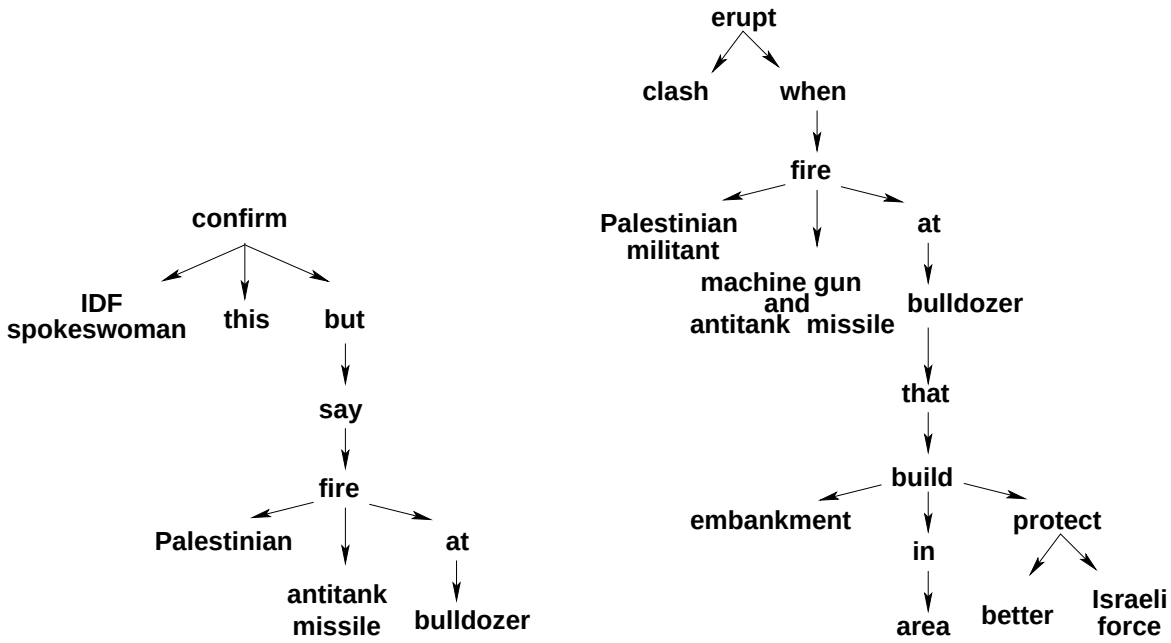
Theme (sentence group) construction: The sentence grouping or theme construction component, SimFinder, used by Barzilay & McKeown (2005) is more complex than ours. It utilizes supervised learning and is trained on a set of linguistic features – e.g., WordNet synsets, syntactic dependencies. However, a simpler algorithm of Nelken & Shieber (2006) which our sentence grouping module draws upon outperforms SimFinder (see Chapter 3).

Theme (group) ranking: Sentence groups are ranked, and the *n* top-scoring ones are chosen for fusion, i.e., from those *n* groups *n* new sentences are later generated. Such factors as the size of the group, the similarity within the group, and the number of lexical chains (Barzilay & Elhadad, 1999) running through the group contribute to the rank. The groups are ordered chronologically, i.e., themes appearing in earlier news articles precede those which appeared later.

Tree transformation: Dependency trees are obtained from phrase structure parses output by Collins's 2003 parser with a set of rules. These are further transformed in several respects: grammatical features and auxiliary nodes are removed from the tree and recorded; noun phrases are flattened. Two examples of transformed trees corresponding to the sentences in (7.1-7.2) (taken from Barzilay & McKeown (2005)) are given in Figure 7.3.

(7.1) IDF Spokeswoman did not confirm this, but said the Palestinians fired an antitank missile at a bulldozer.

(7.2) The clash erupted when Palestinian militants fired machine guns and antitank missiles at a bulldozer that was building an embankment in the area to better protect Israeli forces.



(a) Transformed dependency tree of (7.1) (b) Transformed dependency tree of (7.2)

Figure 7.3: Transformed trees of sentences (7.1-7.2)

(7.3) The army expressed “regret at the loss of innocent lives” but a senior commander said troops had shot in self-defense after being fired at while using bulldozers to build a new embankment at an army base in the area.

Tree alignment and basis tree selection: Pairs of transformed trees from one group are aligned, and each alignment gets its alignment score. The alignment algorithm proceeds in a bottom-up manner and, using dynamic programming, finds locally optimal alignments by taking node similarity as well as structural similarity into account. Figure 7.4 shows the alignment structure of the trees in Figure 7.3. Solid lines represent aligned edges; dotted and dashed lines represent unaligned edges from the trees in Figures 7.3a and 7.3b respectively. Note that the nodes corresponding to *Palestinian* and *Palestinian militants* are aligned although they are not identical. We skip detailed presentation of the algorithm as well as its pseudocode which can be found in the cited article. Here, it is important to stress the following features of the algorithm:

- Several sources of information are used to measure node similarity: lemma identity, lexical relations encoded in WordNet (Fellbaum, 1998) and an automatically extracted paraphrase dictionary.
- In some cases structural similarity enforces alignment of nodes which are neither synonymous nor paraphrases. For example, chances are high that two verbs are aligned

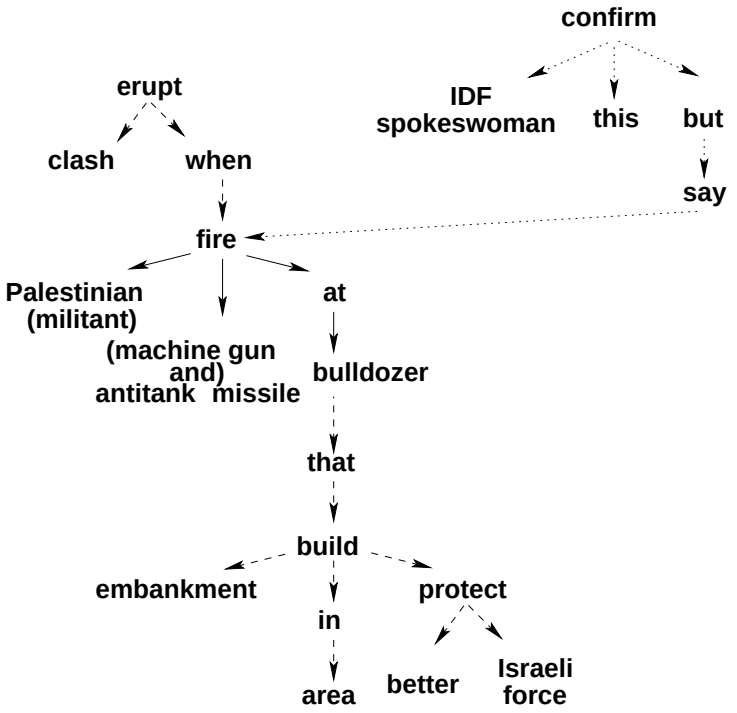
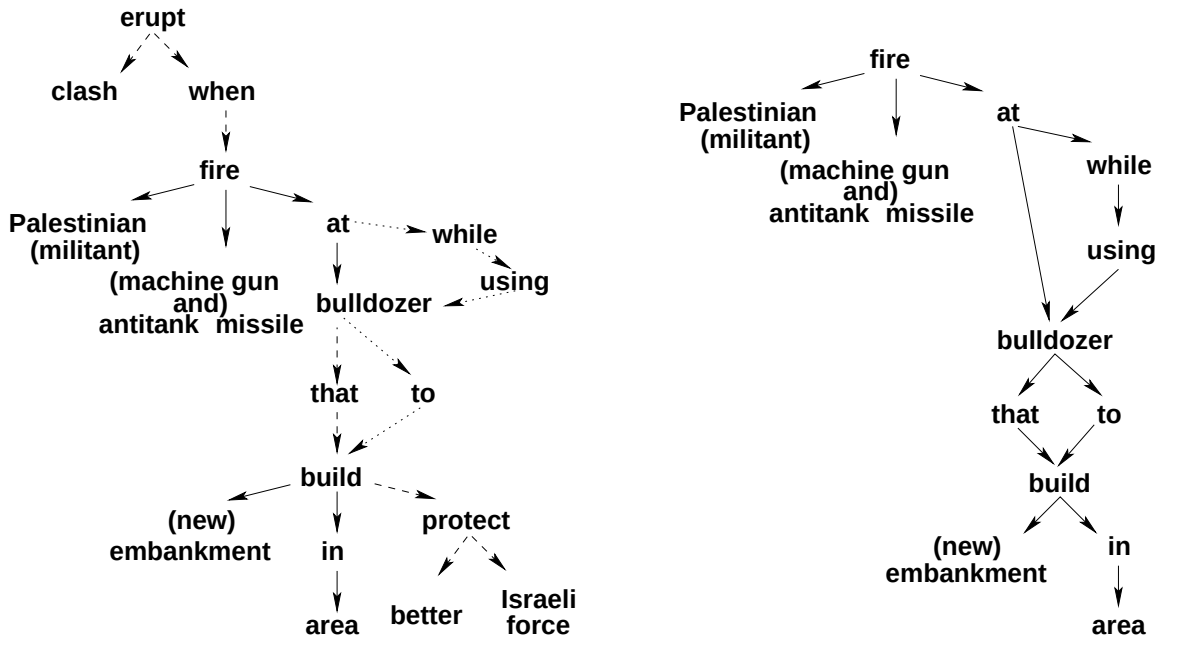


Figure 7.4: Alignment structure of the trees in Figures 7.3a-7.3b

given that their subjects and objects are aligned.

- The algorithm finds a locally optimal alignment – searching for a globally optimal one is NP-hard.
- Once all pairwise alignments and their respective scores have been found, the centroid tree of the group is identified. This tree, called basis tree, has the maximum similarity to other trees in the group.

Basis tree augmentation and pruning: During this stage the basis tree is modified in two ways. Firstly, alternative verbalizations are added. These are taken from the nodes which have been aligned with the nodes in the basis tree. Furthermore, a subtree from a tree other than the basis one is inserted provided that its root is aligned with a node of the basis tree and that the subtree appears in at least half of the sentences from the group. This rule is quite restrictive but reduces the chances of generating ungrammatical or semantically unacceptable sentences. Secondly, certain subtrees are removed provided that they are unaligned. The list of prunable components includes a clause in the clause conjunction, relative clauses, adverbs and prepositional phrases. The transformed tree in Figure 7.3b is the basis tree of the sentence group (7.1-7.3). Figures 7.5a and 7.5b represent the basis tree after augmentation and pruning respectively.



(a) Augmented basis tree

(b) Basis tree after pruning

Figure 7.5: Basis tree from Fig. 7.3b after augmentation and pruning

Linearization: The best linearization of the dependency structure is found by overgeneration and ranking. From the set of all possible strings the one with the lowest length-normalized entropy is selected. The entropy is estimated with a trigram language model trained on a collection of 60M news. The linearizations differ not only in the set of words they cover and word order but also in node realizations (e.g., *new embankment* vs. *embankment*). Information available from the input has been used to reduce the number of possible strings (e.g., the fact that two words are found in a certain order in all the input sentences). Furthermore, the number of linearizations is limited to the first 20,000. The best linearization of the dependency structure in Figure 7.5b is reported to be *Palestinians fired an antitank missile at a bulldozer*.

7.2.3 Reimplementation for German

From the fusion steps described above we reimplemented the following ones:

Tree Transformation: Similarly to the original implementation, we flatten all NPs and remove function words from the representation.

Tree Alignment and Basis Tree Selection: Since the alignment algorithm is language-independent, we implemented it from the pseudocode presented in the original article. We did not use a paraphrase corpus and measured node similarity by lemma identity and with GermaNet

which is an analog of WordNet for German (Lemnitzer & Kunze, 2002). The tree having the maximum average similarity to other trees in its group is selected as the basis tree.

Basis Tree Augmentation and Pruning: The dependency tree is augmented with dependencies shared by at least half of all the input sentences. After augmentation, the verb shared by several trees is found by descending from the root – i.e., the highest shared verb is found. All clauses up this verb node are removed, if there are any, because they appear in the basis tree only. Then the lowest shared verb node is found and all clauses below this verb are removed. After that adverbs and prepositional phrases are removed provided that they are not shared by several input trees.

Linearization: Recall that the dependency structure emerging as a result of basis tree augmentation and pruning is not necessarily a tree (see Fig. 7.5b) and thus cannot be sent to the linearizer directly. To amend this problem, we prepare a method which extracts all possible trees from the dependency structure. An important condition on the trees is that they cover as many nodes as possible. This condition is necessary to put a reasonable limit on the number of possible structures. Each of these trees is then linearized with the combined method. Since the baseline collapses NPs in one node, in most cases the linearizer has to order constituents on the clause level only – the word order within constituents is already given. Finally, given a list of the best linearizations for all the extracted trees, the one with the lowest entropy is selected.

7.3 Online Experiment

We evaluated the three systems – DEFUSER, B&MC and the RANDOM baseline – by means of an online experiment. The total of 50 self-reported native German speakers participated in the experiment. Altogether 120 fused sentences were evaluated. These were generated from 40 randomly drawn groups of related sentences with the three methods (3×40). The participants were asked to read a fused sentence preceded by the input and to rate its readability (*read*) and informativity (*inf*) with respect to the input on a five point scale. The participants were asked to ignore punctuation errors as punctuation is not generated. The experiment was designed so that every participant rated 40 sentences in total. No participant saw two sentences generated from the same input. The experiment was self-paced: the participants proceeded to the next example after they submitted the score for the current one. The invitation email, the instructions in German are presented in Appendix A.1. Figure 7.6 is a screenshot of the evaluation window with a sentence generated by deFuser.

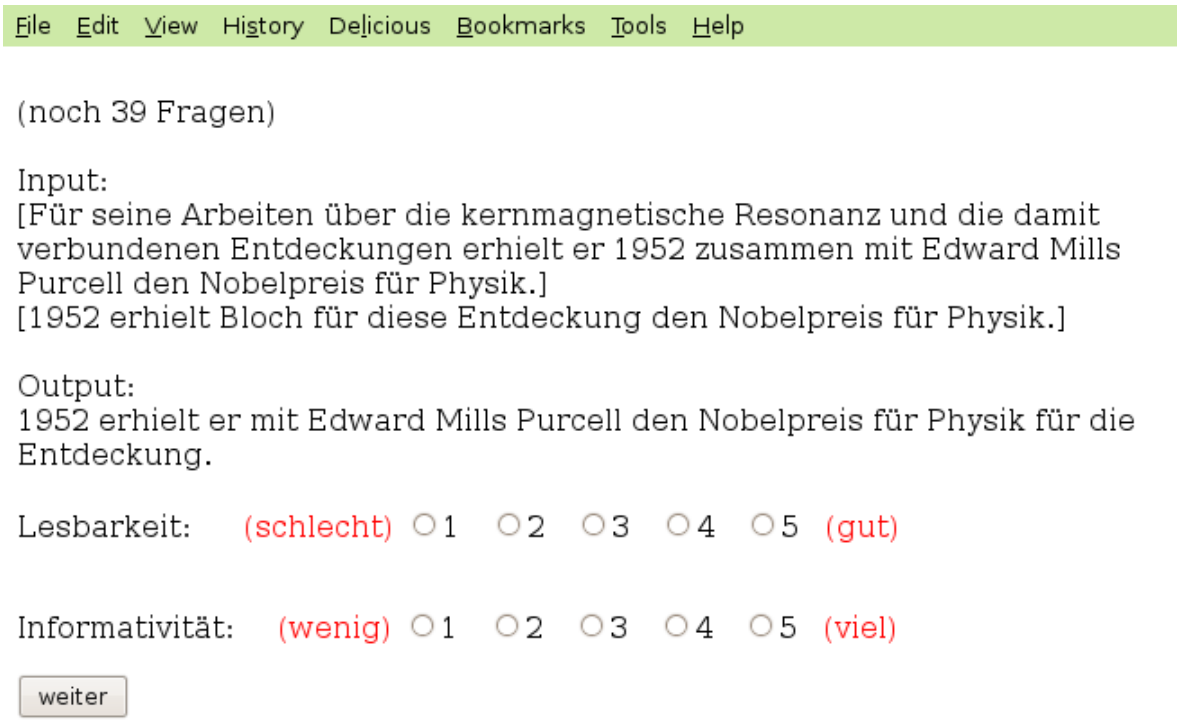


Figure 7.6: Screenshot of the evaluation window

	read	*inf*	*len*
RANDOM	4.0	3.5	12.9
B&MC	3.1	3.0	15.5
DEFUSER	3.7	3.1	13.0

Table 7.1: Average readability and informativity on a five point scale, average length in words

7.4 Results

The results of the online experiment are presented in Table 7.1. The rightmost column, *len*, gives the average length in words of the output of each of the systems. A paired t-test revealed significant differences between the readability ratings of the three systems ($p < 0.01$). The difference between the informativity scores of our system and the baseline (B&MC) is significant with $p < 0.05$ but not with $p = 0.01$.

7.4.1 Error Analysis

The main disadvantage of our method, as well as other methods designed to work on syntactic structures, is that it requires a very accurate parser. In some cases, errors in the preprocessing made extracting a valid dependency tree impossible. The rating of RANDOM demonstrates that errors of the parser and of the linearization module affected the output.

Although the semantic constraints ruled out many anomalous combinations, the limited coverage of GermaNet and the taxonomy derived from Wikipedia was the reason for some semantic oddities in the sentences generated by our method. For example, it generated phrases such as *aus England und Großbritannien* (*from England and Great Britain*). A larger taxonomy would presumably increase both the precision and the recall of the semantic constraints. Such errors were not observed in the output of the baseline because it does not fuse within NPs.

Both B&MC and DEFUSER made subcategorization errors, although these were more common for B&MC. This seems to be due to the fact that it aligns not only synonyms but also verbs which share some arguments. Also, B&MC pruned some PPs necessary for a sentence to be complete. For example, it pruned *an der Atombombe* (*on the atom bomb*) and generated an incomplete sentence *Er arbeitete* (*He worked*). For B&MC, alignment of flattened NPs instead of words caused generating very wordy and redundant sentences when the input parse trees were incorrect.

In a few cases, our method made mistakes in linearizing constituents because it had to rely on a language model whereas the baseline used unmodified constituents from the input. Al-

though the participants were asked to ignore punctuation errors, this sometimes was not easy and absence of intraclause commas caused a drop in readability in some otherwise grammatical sentences.

It is striking that the sentences generated by B&MC turned out to be longer than input ones. This is largely due to the small size of similar sentence groups in a combination with the alignment method which, as we have pointed out before, allowed alignment of dissimilar nodes (in particular, verbs). Given that the average size of a related sentence group is about three, groups of just two sentences were not uncommon. In such cases the augmented tree covered a significant portion of both trees because the minimum frequency threshold for a node to be added to the basis tree was as small as one.

7.4.2 Discussion

There is a considerable difference between the readability and informativity ratings of deFuser. The former is quite close to the upper bound whereas the latter is much lower than the lower bound. Although the poor informativity rating is discouraging, we believe that it is due to the experiment setting and not to the approach itself for the following reasons:

1. Readability of single sentences can be judged out of context and independently from the task as it is an inherent property of the sentence itself. Unlike that, informativity is defined with respect to a given need and may vary from query to query or between users. For example, the very same summary can be informative for one user and totally uninformative for another given the difference in their background and/or queries they made. Therefore, poor readability ratings would imply that the method is unable to generate grammatical and sound sentences, and a change of task would not amend deficiency. On the contrary, poor informativity scores encourage a better defined task rather than prove that the method is of no use.

2. Indeed, in their feedback some participants reported that informativity was hard to estimate. In general it was unassessable for ungrammatical sentences and even for grammatical sentences it was difficult to estimate. This feedback correlates with what Daumé III & Marcu (2004) found in the context of generic single document summarization: generic sentence fusion is an ill-defined task.

3. The main aspiration of our work was to find a way of generating novel grammatical sentences without reliance on hand-crafted rules and expensive resources and without having a single grammatical tree as a basis. From this perspective relatively high readability ratings support our approach which can and should be further improved by refining the content selection part of it.

Compared with the non-trivial baseline, an important advantage of deFuser is that it clearly separates the task of producing a single grammatical syntactic structure from the task of word order generation. Another advantage is that deFuser distinguishes between more and less obligatory arguments. For example, it knows that *at* is more important than *to* for *study* whereas for *go* it is the other way round.

7.5 Summary

In this chapter we presented the results of an online experiment the goal of which was to evaluate overall readability as well as informativity of fused sentences. We compared deFuser with two baselines – a trivial random baseline and a reimplementation of the algorithm of Barzilay & McKeown (2005). The readability results are encouraging being close to the upper bound; the informativity results are poor and need to be verified in an experiment where informativity is better defined. For example, where it is defined with respect to a question, topic, or application.

The relatively high readability rating of our method supports our claim that the method based on syntactic importance score and global constraints generates more grammatical sentences than the previous approach. The average rating of the random baseline provides us with a human evaluation of the constituent ordering method (TWO-STEP) introduced in Chapter 6. The rating is as high as four points (on a five-point scale).

Chapter 8

Sentence Compression with deFuser

As we explained in the introduction, deFuser, in particular the graph compression method presented in Chapter 4, can also be applied to sentence compression. In this case the set of similar sentences consists of exactly one sentence. In this chapter we demonstrate how deFuser, which was initially developed for fusion of German data, can be used for sentence compression of German and English sentences. In Section 8.1 we give a short overview of previous work on sentence compression and in Section 8.2 we explain the benefits of using deFuser for this task. Sections 8.3-8.6 describe how deFuser is applied to sentence compression in English. In the rest of the chapter we present the evaluation and discuss the results.

8.1 Previous Work

Unlike sentence fusion, sentence compression has been explored by many researchers. Below we present an overview of related work classified in two broad categories: approaches which require training data and unsupervised approaches.

8.1.1 Supervised Approaches

Knight & Marcu (2002) apply the noisy-channel model, which had been successfully used in many NLP applications, in particular, machine translation (Brown et al., 1993), to sentence compression. Given a sentence *l* and its parse tree, they search for a compression sentence *s* which maximizes the following probability:

$$s = \arg_s \max P(l|s)P(s)$$

They also introduce a decision-tree model where a compression tree is obtained by making rewriting decisions while processing a parsed sentence.

Riezler et al. (2003) implement a system which operates on LFG structures (Bresnan, 1982) and utilizes an ordered set of *f*-rules which rewrite one *f*-structure into another. The system consists of several components: the LFG parser which produces a set of *f*-structures; the transfer component which operates on those uncompressed structures and produces reduced *f*-structures; a stochastic disambiguator which selects the most well-formed structure with a maximum entropy classifier. Apart from presenting their sentence compression system, Riezler et al. introduce an automatic evaluation metric borrowed from the field of parsing:

$$F\text{-}measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{8.1}$$

where *Precision* and *Recall* are calculated over the sets of labeled dependencies in the compressions generated automatically resp. produced by humans.

Nguyen et al. (2004) describe a system which learns reduction rules (e.g., *shift, reduce, drop*) from an annotated corpus with support vector machines trained on a set of syntactic and semantic and other features.

Turner & Charniak (2005) improve the noisy-channel model of Knight & Marcu (2002) and additionally propose a semi-supervised model for sentence compression. The improvement of the noisy-channel model concerns the use of the syntax-based model of Charniak (2001).

McDonald (2006) introduces a compression system which searches for a compression with maximum scores of adjacent words. The top scoring compression is found with dynamic programming; a rich set of features defined over individual words as well as word bigrams is utilized. One of the strengths of McDonald's approach is that it is robust with respect to noisy features as it learns features' discriminative weights.

Galley & McKeown (2007) also employ a generative model but, unlike Knight & Marcu, calculate $P(s|l)$ directly without breaking it into $P(l|s)P(s)$. The emphasis of their work is on careful feature selection and combination. In particular, they utilize lexical features which make it possible to distinguish cases where a constituent is grammatically obligatory from those where it may be safely deleted.

Cohn & Lapata (2009) present a tree-to-tree rewriting system which learns the weights of rewriting rules and, unlike most compression methods, is able to generate compressions which go beyond word deletion. Similar to Galley & McKeown (2007), Cohn & Lapata observe that lexicalized rules provide better results.

8.1.2 Rule-Based and Unsupervised Approaches

Grefenstette (1998) provides a method of "telegraphic text reduction", which can be classified as a sentence compression method. In his model, the point of compression is to eliminate information of secondary importance based on linguistic criteria. For example, proper names

are judged more important than nouns, which are in turn more important than adjectives, etc. The output text is not necessarily grammatical as, e.g., articles are removed. However, Grefenstette's point is that such a method would be very helpful to summarize text for blind people who cannot quickly skim over a page of text but need to go through it word by word.

Corston-Oliver & Dolan (1999) also shorten sentences albeit with a different goal in mind. They show that the term index can be reduced substantially without dramatically affecting the precision and recall if one omits phrases from subordinate clauses. The motivation for their work comes from Rhetorical Structure Theory (Mann & Thompson, 1988) where certain propositions are judged more important than other. In their experiments, they consider six types of subordinate clauses and observe that their removal does not have a negative effect on the search results.

Unlike the two studies described above, Jing (2000) does not remove phrases based solely on their syntactic categories. Instead, Jing's system removes phrases based on multiple sources of knowledge, including a set of linguistic rules as well a subcategorization lexicon for about 5,000 verbs. Given a parse tree of a sentences, the algorithm first marks for each node whether it is grammatically obligatory based on the knowledge encoded in the rules and in the lexicon. Second, the algorithm assigns each word its context weight, and each non-terminal node gets the sum of the scores of its children as its weight. On the third round, the tree nodes are assigned the probability of being removed which are computed from a corpus of compressions provided by humans (thus, the method is not purely unsupervised). Finally, the annotated tree is traversed in a top-down order and based on the annotation the algorithm decides which nodes should be removed, reduced or retained as they are.

Dorr et al. (2003) generate headlines by compressing first sentences of news articles. Their method utilizes a large set of rules which were induced from headlines produced by humans and which are formulated in terms of syntactic categories and relations.

Gagnon & Da Sylva (2005) present a compression method for French which operates on dependency trees of extracted sentences. They use a considerably large French grammar and devise rules which eliminate verb PP-arguments, subordinate clauses, appositions, etc.

Hori et al. (2003) present an unsupervised approach of speech summarization where each word in an uncompressed sentence is assigned a score and then the compression with the maximum score is found. Several factors contribute to the word score. These are

1. **word significance score** is similar to *tf.idf* and measures the importance of nouns and verbs;

2. **linguistic score** estimates the appropriateness of a string of words and relies on a language model;

3. **confidence score** stands for the confidence of that the acoustic signal has been correctly

recognized;

4. **word concatenation score** is introduced to make sure that the compression is consistent with the input sentence.

The high-scoring compression is found with dynamic programming.

Clarke & Lapata (2006a) present an unsupervised approach which searches for a compression maximizing the language model-based score under a set of grammar constraints. Clarke & Lapata also use a modified version of the word significance score of Hori et al. (2003) which promotes words appearing deeper in the syntactic tree. Further, Clarke & Lapata (2007) extend their model with discourse constraints borrowing from Centering Theory (Grosz et al., 1995) and the theory of lexical cohesion (Halliday & Hasan, 1976; Morris & Hirst, 1991). In their experiments, they demonstrate that taking discourse information into account is beneficial and helps produce compressions which are helpful in the context of question answering.

8.1.3 Discussion of Previous Approaches

The main problem with supervised approaches is the lack of training data which is crucially important for statistical methods. Unlike parallel corpora for machine translation, there exist few manually created compression resources, especially for languages other than English. From this perspective, unsupervised methods are appealing as they hold promise of being portable to languages and domains which lack compression corpora altogether. However, all the unsupervised algorithms we reviewed but the one by Hori et al. (2003) utilize a set of hand-crafted rules when judging the prunability of a phrase. The problem with such rules is that their mastering also requires such resources as human labor or subcategorization lexicons. The less human labor is invested, the more general are the rules and the lower is precision. This becomes evident when rules such as "PPs can be pruned" are analyzed: e.g., in *The president went to Nigeria* the PP *to Nigeria* is arguably as important as the direct object *Nigera* in *The president visited Nigeria*. Thus, truly unsupervised approaches which minimize human interference are of particular value.

8.2 Tree Pruning Approach to Sentence Compression

Incorporating lexical information in supervised systems has been expected to improve performance. However, the bottleneck of supervised approaches is the lack of training data which is required in abundance, especially when lexicalized methods are employed. Since deFuser is an unsupervised system which relies on lexicalized syntactic importance scores calculated from parsed text, it is of interest to see whether such scores help to obtain decent results in sentence compression.

The application of deFuser to sentence compression is straightforward and starts by providing a single sentence as input. Sentence grouping is not required, and the tree alignment step can be skipped too. The semantic constraints are not needed in this case either. However, in Section 8.5 we will show that some other interesting issues related to semantics arise.

Given that deFuser generates a new sentence by first constructing its dependency tree and then linearizing it, it is not limited to performing word deletions only but is also able to do reordering. It should be noted that as it is, deFuser cannot generate lexical or structural paraphrases such as passivization. In this regard it does not differ from the majority of existing systems. However, it is reasonable to expect that in many cases a good compression corresponds to a subtree of the dependency tree of the input. Indeed, it is unlikely that the parse tree of a compression differs considerably from the source tree. For example, it is hard to think of a situation where a prepositional phrase is promoted to the subject role in a compressed sentence. From this perspective the sentence compression task can be viewed as a tree pruning one.

In the following we will present the modules which are necessary for sentence compression, most of which have been introduced in Chapters 4 and 6. We will show that even though we make the simplified assumption that the syntactic importance of an argument depends only on its syntactic label and the lemma of the parent, the compressed sentences obtained this way can compete with the best approaches existing to date (see Sec. 8.7).

8.3 Tree Transformation

Before a dependency tree is compressed, i.e. before some of its dependencies are removed, the tree is modified. We will demonstrate the effect of the transformations with the following English sentence:

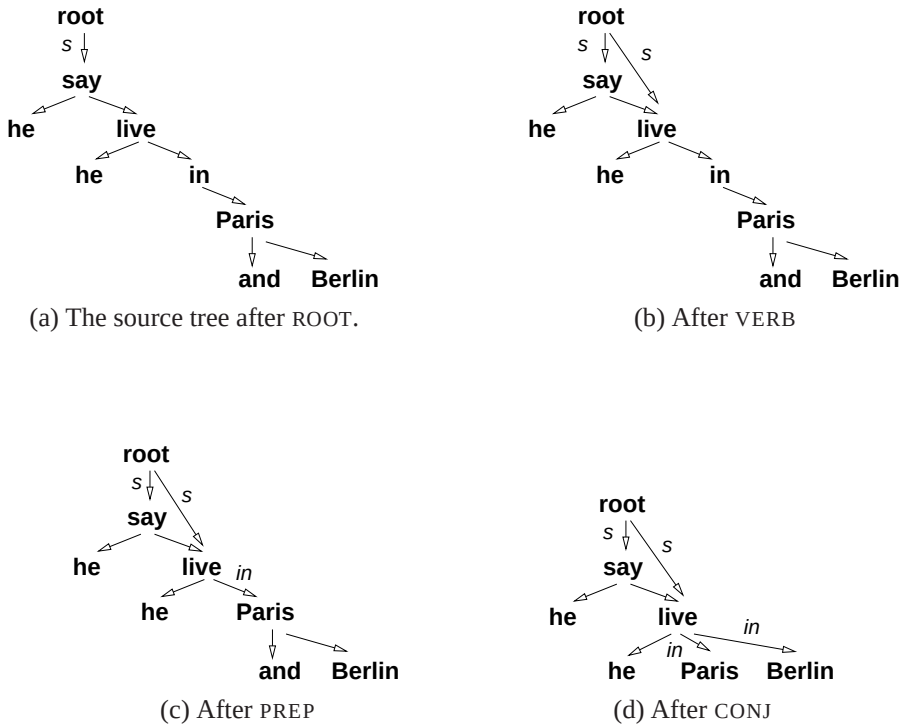
(8.2) He said that he lived in Paris and Berlin

ROOT: an explicit rootnode is inserted and connected with every inflected verb in the sentence. All edges originating from the rootnode bear the *s* label (Fig. 8.1a);

VERB: auxiliary edges are removed and such grammatical properties as voice, tense or negation are memorized for verbs (Fig. 8.1b);

PREP: prepositional nodes are removed and placed as labels on the edge from their head to the respective NP (Fig. 8.1c);

CONJ: a chain of conjoined elements is decomposed and each of them is attached to the head of the first element in the chain with the syntactic label of the first element in the chain (Fig. 8.1d). This transformation is not applied to verbs.



(a) The source tree after ROOT.

(b) After VERB

(c) After PREP

(d) After CONJ

Figure 8.1: The transformations applied to the dependency structure of *He said that he lived in Paris and Berlin* after the transformations

The purpose of the latter two transformations is to make relations between open-class words more explicit. Similar to how we proceeded in Chapter 4, we want to decide whether an edge should be retained or can be pruned judging from two questions: (i) How important is this argument for the head? (ii) How informative is the dependent word? As an example, consider a source sentence given in (8.3). Here, we want to decide whether one prepositional phrase (or both) can be pruned without making the resulting sentence ungrammatical.

(8.3) After some time, he moved to London.

It would not be very helpful to check whether an argument attached with the label *pp* is obligatory for the verb *move*. Looking at a particular preposition (*after* vs. *to*) would be more enlightening. With CONJ we are free to retain any of the conjoined elements in the compression and do not have to preserve the whole sequence of them if we are interested in only one.

Although the transformed dependency structure is not necessarily a tree because of the ROOT transformation, in the following we will refer to it as tree. This is done to distinguish between graphs corresponding to a single tree and those described in Chapter 4 which emerge as a result of merging several trees.

subj	dobj	in	at	after	with	to
0.51	0.8	0.07	0.05	0.01	0.03	0.01

(a) Probabilities of *subj, d(irect)obj, in, at, after, with, to* given the verb *study*

subj	obja	in	an	nach	mit	zu
0.88	0.74	0.44	0.42	0.09	0.02	0.01

(b) Probabilities of *subj, obja(ccusative), in, at, after, with, to* given the verb *studieren*

Table 8.1: Arguments of *study* and *studieren* and their probabilities

8.4 Tree Compression

Similarly to what we have done before, we formulate the compression task as an optimization problem which we solve using integer linear programming. Given a transformed dependency tree, we decide which dependency edges to remove. For each directed dependency edge from head h to word w we thus introduce a binary variable $x^l_{h,w}$ where *l* stands for the edge's label:

$$x^l_{h,w} = \begin{cases} 1 & \text{if the dependency is retained} \\ 0 & \text{otherwise} \end{cases} \tag{8.4}$$

The goal is to find a subtree which gets the highest score of the objective function (8.5) to which both the probability of dependencies ($P(l|h)$) and the importance of dependent words ($I(w)$) contribute:

$$f(X) = \sum_x x^l_{h,w} P(l|h) I(w) \tag{8.5}$$

Intuitively, the conditional probabilities prevent us from removing obligatory dependencies from the tree. For example, $P(subj|work)$ is higher than $P(with|work)$, and therefore the subject will be preserved whereas the prepositional label and thus the whole PP can be pruned. This way we do not have to create an additional constraint for every obligatory argument (e.g., subject or direct object). Neither do we require a subcategorization lexicon to look up which arguments are obligatory for a certain verb. Verb arguments are preserved because the dependency edges, with which they are attached to the head, get high scores. Tables 8.1a and 8.1b present the probabilities of a number of labels given that the head is *study* and its German counterpart, respectively.

Note that if we did not apply the PREP transformation we would not be able to distinguish between different prepositions and could only calculate $P(pp|studieren)$ which would not be very informative. The probabilities for English are lower than those for German because we calculate the conditional probabilities given word lemma. In English, the part of speech information cannot be induced from the lemma and thus the set of possible labels of a node is on average larger than in German.

Since in English in some cases the lemmas of a verb and a noun coincide, we use the part-of-speech information for disambiguation and compute the probability of a label conditioned on the lemma and the part-of-speech tag of the head ($C(x, y)$ stands for the number of times x and y co-occur):

$$P(l|h) = P(l|h, t_h) = \frac{C(h, t, l)}{C(h, t)} \tag{8.6}$$

For example, $P(of|study, noun) = 0.26$ whereas $P(of|study, verb)$ is undefined.

There are many ways in which word importance $I(w)$ can be defined. Here, we use the formula introduced by Clarke & Lapata (2008) which is a modification of the significance score of Hori et al. (2003):

$$I(w) = \frac{l}{N} f_w \log \frac{F_A}{F_w} \tag{8.7}$$

w is the topic word (either noun or verb), f_w is the frequency of w in the document, F_w is the frequency of w in the corpus, and F_A is the sum of frequencies of all topic words in the corpus. l is the number of clause nodes above w and N is the maximum level of embedding of the sentence w belongs to.

The objective function is subject to constraints of two kinds. The constraints of the first kind are structural and ensure that the preserved dependencies result in a tree. (8.8) ensures that each word has one head at most. (8.9) ensures connectivity in the tree. (8.10) restricts the size of the resulting tree to α words.

$$\forall w \in W, \sum_{h,l} x^l_{h,w} \leq 1 \tag{8.8}$$

$$\forall w \in W, \sum_{h,l} x^l_{h,w} - \frac{1}{|W|} \sum_{u,l} x^l_{w,u} \geq 0 \tag{8.9}$$

$$\sum_{x} x^l_{h,w} \leq \alpha \tag{8.10}$$

α is a function of the length of the source sentence in open-class words. The function is not linear since the degree of compression increases with the length of the sentence. The compression rate of human-generated sentences is about 70% (Clarke & Lapata, 2008)[1]. To

[1] Higher rates correspond to longer compressions.

approximate this value, we set the proportion of deleted words to be 20% for short sentences (5-9 non-stop words) and up to 50% for long sentences (30+ words).

The constraints of the second type ensure the syntactic validity of the output tree and explicitly state which edges should be preserved. These constraints can be general as well as conditional. The former ensure that an edge is preserved if its source node is retained in the output. Conditional syntactic constraints state that an edge has to be preserved if (and only if) a certain other edge is preserved. We have only one syntactic constraint which states that a subordinate conjunction (*sc*) should be preserved if and only if the clause it belongs to functions as a subordinate clause (*sub*) in the output. If it is taken as the main clause, the conjunction should be dropped. In terms of edges, this can be formulated as follows (8.11):

$$\forall x^{sc}_{w,u}, x^{sub}_{h,w} - x^{sc}_{w,u} = 0 \tag{8.11}$$

Due to the constraint (8.8), the compressed subtree is always rooted in the node added as a result of the first transformation. A compression of a sentence to an embedded clause is not possible unless one preserves the structure above the embedded clause. Often, however, main clauses are less important than an embedded clause. For example, given the sentence in (8.12) it is the embedded clause which is informative and to which the source sentence should be compressed.

(8.12) He said they have to be held in Beirut

The purpose of the ROOT transformation is to amend exactly this problem. Having an edge from the rootnode to every inflected verb allows us to compress the source sentence to any clause. Thus, the method we have presented so far is a reduced version of the graph compression method developed for sentence fusion. We omit the semantic constraints as the co-arguments' compatibility problem is unlikely to arise in this setting. However, in the next section we address another issue concerning semantics.

8.5 Retaining Noun Modifiers

While testing deFuser on some of the 2008 TAC[2] and the compression corpus (see Sec. 2.2.1) data, among other mistakes we discovered a class of cases where the compression had a meaning quite different from the meaning of the source sentence. Consider the following real data examples and their automatic compressions:

(8.13) He died late Wednesday after a long battle with cancer, his cousin said.

[2] See the Text Analysis Conferences website `http://www.nist.gov/tac`; before 2008 Document Understanding Conferences `http://duc.nist.gov`.

(8.14) He died after a battle.

(8.15) The ban supports an anti-sweets campaign by the Paediatrics Society of Thailand to reduce the numbers of children hooked on sugar, which can lead to diabetes and obesity.

(8.16) The ban supports an anti-sweets campaign to reduce the numbers of children.

Sentences in (8.13) and (8.15) are sentences from the TAC[3] and the compression corpora[4], respectively; sentences (8.14) and (8.16) are compressions generated by our system. Apparently, in both cases the meaning of the compression is very different from the meaning of the source sentence. Generally speaking, the problem can be attributed to the fact that noun modifiers, crucially important for a correct interpretation, were removed. In the case of (8.14), this led to a literal interpretation of the word *battle* which was used metaphorically in (8.13). In the case of (8.16), the object of the verb *reduce* became all the children of Thailand while in the source sentence only a very particular subgroup of the children was referred to.

One possible way of amending the "missing-modifiers" problem is to look at the context where the sentences are found and based on that decide which words are particularly related to the context. This approach is undertaken by Clarke & Lapata (2007) who integrate lexical chains (Barzilay & Elhadad, 1999) and Centering Theory (Grosz et al., 1983) into their compression system to enrich it with discourse information. Indeed, (8.15) is from a set of news collected for the query *Describe actions that are being taken to decrease childhood obesity*, and such words as *diabetes, obesity, sweets, sugar, diet*, etc. are very frequent in the news. However, (8.13) is found in the document which is not about cancer, but about a person who died from it. In that document, there are hardly any words related to cancer, therefore taking local context into account is unlikely to retain the modifier *with cancer* of the noun *battle*. Moreover, the reason for why the modifier is important seems to be different there. Consider the following discourse:

(8.17) He died late Wednesday after a long battle with cancer, his cousin said. The battle with cancer lasted for several years and completely exhausted him.

Unlike (8.13), in the second sentence in (8.17) the PP *with cancer* can be pruned without distorting the meaning of the sentence because the definite NP *the battle* would refer unambiguously to the previously mentioned *battle with cancer*. Thus, the problem is not that the relation between the modifier and the previous discourse is not recognized but rather has to do with the referential clarity of the noun phrase.

[3] See D0825E in the test collection.

[4] See file latwp970717.0099 in the compression corpus.

The last observation motivates a semantic constraint whose purpose is preserve at least one of the modifiers of a common noun which appears either in singular with an indefinite article, or in plural without a determiner. More formally it can be stated as a constraint:

$$\forall h : (\texttt{pos}(h) = \texttt{nn} \wedge \texttt{det}(h) \in \{\texttt{a}, \texttt{an}\}) \vee (\texttt{pos}(h) = \texttt{nns} \wedge \texttt{det}(h) = \emptyset), \sum_{l,w} x^l_{h,w} \geq 1 \quad (8.18)$$

In the experiments section (Sec. 8.7), we discuss the effect this constraint had on the overall performance of the compression system.

8.6 Tree Linearization

A very simple but reasonable linearization technique is to present the words of a compressed sentence in the order they are found in the source sentence. This method has been applied before and this is how we linearize the trees obtained from the English data. Unfortunately, this method cannot be directly applied to German because of the constraints on word order in this language. One of the rules of German grammar states that in the main clause the inflected part of the verb occupies the second position, the first position being occupied by exactly one constituent (recall the constraint on verb placement from Sec. 5.1). If the sentence initial position in a source sentence is occupied by a constituent which got pruned off as a result of compression, the verb becomes the first element of the sentence which results in an undesirable output. Therefore, we use the linearization method introduced in Chapter 6 to convert compressed trees into German sentences.

8.7 Experiments

In this section we present the results of a compression experiment. We used the automatically annotated compression corpus (see Sec. 2.2.1) and tested the system on the complete set of 82 documents. The frequencies needed for the syntactic and word informativeness scores were precomputed on the automatically annotated WSJ corpus (described in Sec. 2.2.2). In particular, the word informativeness score was calculated on the nouns and verbs from WSJ. Conditional probabilities for the syntactic score were calculated from a smaller portion of WSJ (about 6 million tokens). The latter number is comparable to the size of the dataset we used to compute the probabilities for German. For German, we use the Wikipedia Corpus (about 4,000 articles) as well as TüBa-D/Z (see Sec. 2.1) to calculate conditional probabilities and significance scores. The frequency counts are then updated with the new compression data (see Sec. 4.4.3.3). Thus, our algorithm does not require smoothing as there are no unseen words and dependencies.

	F-measure	*compr.rate*
LM+SIG+CONSTR	40.5	72.0%
DEP. TREES (RASP)	40.7	49.6%
DEP. TREES (SP)	50.4	66.3%
GOLD	-	72.1%

Table 8.2: Average results on the English corpus

8.7.1 Evaluation and Results

We evaluate the results automatically as well as with human subjects. In our evaluation we relied on previous research and followed the methodology of Clarke & Lapata (2008).

8.7.1.1 Automatic Evaluation (English)

To automatically assess the performance of the method, we calculate the F-measure on grammatical relations. Following Riezler et al. (2003), we calculate average precision and recall as the amount of grammatical relations shared between the output of our system and the gold standard variant divided over the total number of relations in the output and in the human-generated compression, respectively (see (8.1)). According to Clarke & Lapata (2006b), this measure reliably correlates with human judgments. The results of our evaluation as well as the result by a recent compression system evaluated on the same corpus are presented in Table 8.2. The system (LM+SIG+CONSTR) developed by Clarke & Lapata (2008) used language model scoring (LM), word significance score (SIG), and linguistic constraints (CONSTR).

The F-measure reported by Clarke & Lapata (2008) is calculated with RASP although the system of the latter builds upon a different parser. For our system we present the results obtained on the data parsed with RASP as well as with the Stanford parser (SP). In both cases the F-measure is found with RASP in order to allow for a fair comparison between all the systems. We recalculate the compression rate for the gold standard ignoring punctuation. On the test portion of the corpus the compression rate turns out to be slightly higher than that reported by Clarke & Lapata (2008) (70.3%).

8.7.1.2 Evaluation with Native Speakers (German)

As there are no human-generated compressions for German data, we evaluate the performance of the method in terms of **grammar** (grammaticality) and **importance** by means of an experiment with native speakers. Importance represents the amount of relevant information from the source sentence retained in the compression. In the experiment, subjects are presented with a source sentence and its compression which they are asked to evaluate on two five-point scales.

	grammar	*importance*
LM+SIG+CONSTR	3.76	3.53
DEP. TREES (DE)	3.62	3.21

Table 8.3: Average results for the German data

Higher grades are given to better sentences. Since our method does not generate punctuation, the judges are asked to ignore errors due to missing commas. Five participants took part in the experiment and each rated the total of 25 sentences originating from a randomly chosen newspaper article from TüBa-D/Z (see Sec. 2.1.3). Their ratings as well as the ratings reported by Clarke & Lapata (2008) on the compression corpus in English are presented in Table 8.3.

8.7.2 Discussion

The results on the English data are comparable[5] with the results by Clarke & Lapata (2008). The comparable performance of our system was achieved with a single syntactic constraint (8.11) and without any elaborated resources. This gives us a promise that our system is adaptable to other languages.

The results on the German set are not conclusive since the number of human judges is quite small. Still, these preliminary results are comparable to those reported for English and thus give us some evidence that the method can be adapted to languages other than English. Interestingly, the importance score depends on the grammaticality of the sentence. A grammatical sentence can convey unimportant information but it was never the case that an ungrammatical sentence got a high rating on the importance scale. Some of the human judges told us that they had difficulties assigning the importance score to ungrammatical sentences.

Bellow we discuss the impact of the parser choice on the performance and the linguistic constraint we introduced in Section 8.5, as well as say a few words about the automatic evaluation.

8.7.2.1 Impact of the Parser Choice

In order to explain why the performance depends so much on the parser, we calculated the dependency-based compression precision P of the compressions produced by humans. This is defined in (8.19) as the number of dependencies shared by a human-generated compression (dep_c) and the source sentence (dep_s) divided over the total number of dependencies found in

[5] They are clearly better than those reported by Clarke & Lapata (2008) and presented here but are worse than the results of a replicated experiment done by Cohn & Lapata (2009) which are above 50%.

	RASP	Stanford parser
precision	79.6%	84.3%

Table 8.4: Precision of the parsers

the compression:

$$P = \frac{|dep_c \cap dep_s|}{|dep_c|} \tag{8.19}$$

The intuition is that if a parser does not reach high precision on gold standard sentences, i.e., if it does not assign similar dependency structures to a source sentence and its compression, then it is hopeless to expect it to produce good compression with our dependency-based method. Indeed, the precision should be high as it is unlikely for two words to change their dependency type or cease to be in the dependency relation as a result of compression. It is difficult to think of an example where a word, which attaches to the verb with the label *subj*, gets a new head or another label in the compression. However, the precision does not have to be as high as 100% because of, e.g., changes within a chain of conjoined elements or appositions. The precision of the two parsers calculated over the compression corpus is presented in Table 8.4.

The precision of the Stanford parser is about 5% higher than that of RASP. In our opinion, this partly explains why the use of the Stanford parser increases the F-measure by nine points. Another possible reason for this improvement is that the Stanford parser identifies three times more dependency relations than RASP and thus allows for finer distinctions between the arguments of different types.

8.7.2.2 Impact of the Noun Modifiers Constraint

To assess the impact of the noun modifiers constraint introduced in Section 8.5, we measured the performance with and without the constraint and looked at the differences in compressions. The system without the constraint performed better, the difference in F-measure turned out to be of about one point. This is a disappointing result as we naturally hoped to improve performance by keeping crucially important noun modifiers. An error analysis of a random sample of compressed sentences revealed that retaining modifiers caused elimination of a syntactically more important element. However, in certain cases the constraint prevented generation of a sentence inconsistent with the input.

8.7.2.3 Relation between F-measure and Compression Rate

Another point to discuss concerns the compression rates and the relation between compression rate and F-measure. The compressions generated with RASP are considerably shorter than

those generated with the Stanford parser. This is mainly due to the fact that the structure output by RASP is not necessarily a tree or a connected graph. In such cases only the first subtree of the sentence is taken as input and compressed.

The worse performance of RASP-based system in comparison with the Stanford parser-based one can be accounted for by the lower accuracy of RASP, as we have shown (Sec. 8.7.2.1). However, it can also be attributed to a lower compression rate[6]. Indeed, it seems that F-measures should not be compared on their own but only if they are obtained with identical or nearly identical compression rates. For example, a trivial baseline which does not compress sentences at all has a compression rate of 100% and an F-measure as high as 61.4% (average precision and recall are 53.9 and 74.0, respectively).

8.8 Summary

In this chapter we presented an unsupervised method which compresses dependency trees and not strings of words. We have argued that our formulation has the following advantages: firstly, the approach is unsupervised, the only requirement being that there is a sufficiently large corpus and a dependency parser available. Secondly, it requires neither a subcategorization lexicon nor hand-crafted rules to decide which arguments are obligatory. Thirdly, it finds a globally optimal compression by taking syntax as well as word importance into account.

[6]Note that lower rate, somewhat unintuitively, corresponds to shorter sentences.

Chapter 9

Conclusions

The popularity of text summarization has been steadily increasing in recent years. This is not surprising given its practical utility: multi-document summarization systems would be of great use given the enormous amount of information daily appearing online. In the introduction we demonstrated how multi-document text summarization could benefit from sentence fusion. In particular, we argued that the wide-spread extractive strategy falls short on resolving the non-redundancy/completeness problem and that a system capable of generating new sentences directly from text would be very useful. We gave an overview of existing summarization systems which go beyond extraction and pointed out their shortcomings. Furthermore, we showed that to date there is no implemented method which would fulfill all of the following requirements:

- generate complete, grammatical sentences;
- produce sentences which are not only grammatical but also consistent with the input;
- make the length of generated sentences adjustable so that succinct as well as detailed sentences could be produced, dependent on the application.

This thesis presented a novel sentence fusion system, deFuser, developed for and tested on German data which fulfills the above mentioned requirements. Given a collection of related documents, deFuser identifies and groups similar sentences. It combines relevant information from different sources and produces new grammatical sentences; it can also be used for sentence compression. These two properties makes deFuser an attractive tool for multi-document summarization. In this final chapter we summarize the main contributions of our work and outline possible directions for future research.

9.1 Main Contributions

Syntactic and semantic well-formedness. deFuser generates novel sentences by, first, building a grammatical and semantically sound syntactic structure, and second, by finding the best word order for this structure. With regard to these two main steps, it is important to emphasize the following contributions:

1. Grammaticality of the dependency tree is ensured with a few language-independent constraints – neither hand-crafted rules, nor resources unavailable for most languages are required. Relative grammatical importance is determined from a parsed corpus and is used not only for verb arguments but for all words and dependency relations.

2. Apart from grammaticality, which concerns the syntactic side of sentence well-formedness only, deFuser implements a semantic component and checks semantic compatibility of co-arguments. Such a semantic module is missing in most text-to-text generation architectures which use semantic information for similarity identification only.

3. We introduced a novel tree linearization method with considerably less overgeneration than previous approaches. On German data our method outperformed several baselines. We argued that a combined method which separately finds the best order of constituents on the clause level and the best word order within constituents on the phrase level is beneficial. Our experiments confirmed that trigram language models are appropriate for phrase linearization but are not useful on the clause level where long-distance dependencies are typical and more linguistic information is required. We showed that our combined method works equally well on English as well as on German data.

More abstractive fusion. In contrast to previous sentence fusion systems, deFuser abstracts from single sentences to a global representation – a graph of dependencies. This graph is built from transformed syntactic dependencies and reveals semantic relations. It is this semantically and syntactically motivated representation from which deFuser generates new sentences. This is important, because it brings summarization one step closer to abstraction. As a consequence, generation of sentences which incorporate information from different sources becomes possible. Abstracting from sole dependency trees to one global graph also makes the generation task more challenging because a single correct tree is no longer available in the graph.

Applicability to Sentence Compression. We demonstrated that deFuser can also be applied to sentence compression. Tested on English and German data, it showed performance comparable to recently developed systems. Importantly, unlike the majority of sentence compression methods, deFuser is unsupervised and requires an accurate dependency parser only.

Portability. The fact that the amount of hand-crafted rules and the use of expensive resources are minimized, makes the method portable to other languages. The main requisites are (i) a dependency parser, (ii) a taxonomy extracted from Wikipedia, (iii) a sufficiently large corpus from which dependency scores can be calculated. Languages with rich morphology would benefit from a morphological module encoding gender, case and number agreement.

Local coherence in German. We also presented a study on the relation between word order and local coherence in German. The hypothesis we formulated in terms of information structure and discourse status accounts for a range of phenomena in German which could not be explained in terms of discourse status or syntactic relations alone. A corpus analysis and an experiment with native speakers supported our hypothesis which laid a basis for the tree linearization algorithm.

9.2 Further Research

An important property of deFuser is that it can be extended with relatively little effort due to the constraint-based framework we adopted. In particular, the following directions of future research seem worth investigating:

Summary generation. deFuser has been designed to generate single sentences. When combined together, generated sentences might produce a poorly structured and locally incoherent discourse. While we believe that discourse structuring should be performed by an extra module before sentences are generated, enforcing local coherence could and very probably should be ensured during sentence generation. In the present work we investigated one aspect of local coherence related to word order. Another aspect could be the proper choice of referring expressions. In an architecture where a text is generated incrementally – i.e., sentence by sentence – constraints on the use of pronouns/full noun phrases could be formulated. For example, given that a new entity is introduced, the use of pronouns should be avoided. Similarly, one may formulate constraints preventing repetitions.

Topic-oriented summarization. Throughout the thesis claims have been made that deFuser can be adapted to topic-oriented summarization. Indeed, novel trees are constructed with regard to two considerations – grammaticality and informativeness. Apart from the syntactic importance of a dependency, the informativeness of the word the dependency points to has been taken into account. So far we have used a formula for generic summarization which promotes words frequent in the input but infrequent in our corpora. Apparently, this is only one of the many ways informativeness could be defined. In the future, it would be of interest

to see how well the weighting schemes designed for topic-oriented summarization can be "plugged-in" into deFuser and evaluate the system in a topic-driven setting.

Question answering. Question answering is another subfield of natural language processing where generation of novel sentences is required. Like in summarization, there it is also important to produce a focused sentence which does not convey irrelevant information. Given a set of sentences answering a question specified by the user, deFuser would aim at constructing a dependency tree such that it retains the words which are part of the answer. Hard constraints could be used to retain relevant words. The goal of deFuser would be to find the minimum grammatical tree covering those words.

Adaptation to other languages. The languages we have so far experimented with are English and German. Despite a number of important differences, the two are very similar especially when compared with members from the Semitic or Slavic language families. For example, Czech has a number of well-developed resources – annotated corpora, dependency parsers – and thus it would be of interest to see how well deFuser would perform on this language. This is particularly interesting given that there are hardly any summarization systems implemented for languages other than English.

Appendix A

Online-Experiment Instructions

A.1 Invitation to Participate in the Experiment

Hallo,

ich möchte Euch alle als deutsche Muttersprachler bitten, an einem kleinen Evaluierungsexperiment teilzunehmen. Was Ihr bewerten werdet, sind Sätze, die nach drei verschiedenen Methoden aus jeweils zwei-vier gegebenen Sätzen generiert wurden. Die neuen Sätze sollen auf Lesbarkeit und Informationsgehalt bewertet werden. Das Experiment sollte weniger als 20 Minuten dauern und ist anonym. Eure Hilfe brauche ich bis spätestens Donnerstag Mittag.

Das Experiment findet Ihr unter

`http://212.126.215.106/fusion`

Vielen Dank im Voraus!

Liebe Grüße,

Katja

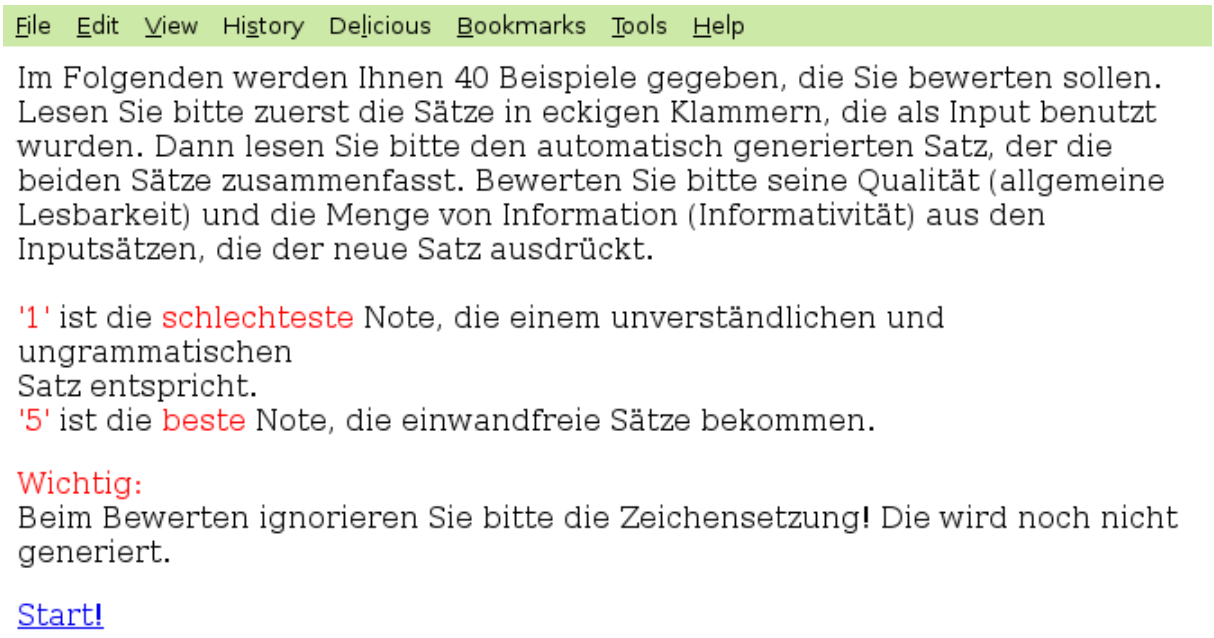


Figure A.1: Screenshot of the instruction window

A.2 Instructions

Im Folgenden werden Ihnen 40 Beispiele gegeben, die Sie bewerten sollen. Lesen Sie bitte zuerst die Sätze in eckigen Klammern, die als Input benutzt wurden. Dann lesen Sie bitte den automatisch generierten Satz, der die beiden Sätze zusammenfasst. Bewerten Sie bitte seine Qualität (allgemeine Lesbarkeit) und die Menge von Information (Informativität) aus den Inputsätzen, die der neue Satz ausdrückt.

> ’1’ ist die schlechteste Note, die einem unverständlichen und ungrammatischen Satz entspricht.
> ’5’ ist die beste Note, die einwandfreie Sätze bekommen.

Wichtig:
Beim Bewerten ignorieren Sie bitte die Zeichensetzung! Die wird noch nicht generiert.

`Start!`

Bibliography

Althaus, E., N. Karamanis & A. Koller (2004). Computing locally coherent discourses. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, Barcelona, Spain, 21–26 July 2004, pp. 400–407.

Banko, M., V. O. Mittal & M. J. Witbrock (2000). Headline generation based on statistical translation. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, Hong Kong, China, 1–8 August 2000, pp. 318–325.

Barzilay, R. & M. Elhadad (1999). Using lexical chains for text summarization. In I. Mani & M. T. Maybury (Eds.), *Advances in Automatic Text Summarization*, pp. 111–121. Cambridge, Mass.: MIT Press.

Barzilay, R. & M. Elhadad (2003). Sentence alignment for monolingual comparable corpora. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, Sapporo, Japan, 11–12 July 2003, pp. 25–32.

Barzilay, R., N. Elhadad & K. R. McKeown (2002). Inferring strategies for sentence ordering. *Journal of Artificial Intelligence Research*, 17:35–55.

Barzilay, R. & M. Lapata (2006). Aggregation via set partitioning for natural langauge generation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, New York, N.Y., 4–9 June 2006, pp. 359–366.

Barzilay, R. & K. R. McKeown (2005). Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–327.

Berger, A., S. A. Della Pietra & V. J. Della Pietra (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Branavan, S., P. Deshpande & R. Brazilay (2007). Generating a table-of-contents. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, Prague, Czech Republic, 23–30 June 2007, pp. 544–551.

Brants, T. (2000). TnT – A statistical Part-of-Speech tagger. In *Proceedings of the 6th Conference on Applied Natural Language Processing*, Seattle, Wash., 29 April – 4 May 2000, pp. 224–231.

Bresnan, J. (1982). *Mental Representation of Grammatical Relations*. Cambridge, Mass.: MIT Press.

Briscoe, E., J. Carroll & R. Watson (2006). The second release of the RASP system. In *Proceedings of the COLING-ACL Interactive Presentation Session*, Sydney, Australia, 2006, pp. 77–80.

Brown, P. F., S. A. Della Pietra, V. J. Della Pietra & R. L. Mercer (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Cahill, A. & M. Forst (2009). Human evaluation of a german surface realisation ranker. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, Athens, Greece, 30 March – 3 April 2009 , pp. 112–120.

Cahill, A., M. Forst & C. Rohrer (2007). Stochastic realization ranking for a free word order language. In *Proceedings of the 11th European Workshop on Natural Language Generation*, Schloss Dagstuhl, Germany, 17–20 June 2007, pp. 17–24.

Carbonell, J. G. & J. Goldstein (1998). The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, Melbourne, Australia, 24–28 August 1998, pp. 335–336.

Chafe, W. (1976). Givenness, contrastiveness, definiteness, subjects, topics, and point of view. In C. Li (Ed.), *Subject and Topic*, pp. 25–55. New York, N.Y.: Academic Press.

Chafe, W. (1987). Cognitive constraints on information flow. In R. S. Tomlin (Ed.), *Coherence and Grounding in Discourse*, pp. 21–52. Amsterdam, The Netherlands: John Benjamins.

Charniak, E. (2001). Immediate-head parsing for language models. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, Toulouse, pp. 116–123.

Clarke, J. & M. Lapata (2006a). Constraint-based sentence compression: An integer programming approach. In *Proceedings of the Poster Session at the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, Sydney, Australia, 17–21 July 2006, pp. 144–151.

Clarke, J. & M. Lapata (2006b). Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, Sydney, Australia, 17–21 July 2006, pp. 377–385.

Clarke, J. & M. Lapata (2007). Modelling compression with discourse constraints. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Language Learning*, Prague, Czech Republic, 28–30 June 2007, pp. 1–11.

Clarke, J. & M. Lapata (2008). Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:399–429.

Clarkson, P. & R. Rosenfeld (1997). Statistical language modeling using the CMU-Cambridge toolkit. In *Proceedings of the 5th European Conference on Speech Communication and Technology*, Rhodes, Greece, 22-25 September 1997, pp. 2707–2710.

Cohn, T. & M. Lapata (2009). Sentence compression as tree transduction. *Journal of Artificial Intelligence Research*, 34:637–674.

Collins, M. (2003). Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

Cormen, T. H., C. E. Leiserson, R. L. Rivest & C. Stein (2001). *Introduction to Algorithms*. MIT Press and McGraw-Hill.

Corston-Oliver, S. H. & W. B. Dolan (1999). Less is more: eliminating index terms from subordinate clauses. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, College Park, Md., 20–26 June 1999, pp. 349–356.

Daumé III, H. & D. Marcu (2002). A noisy-channel model for document compression. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, Penn., 7–12 July 2002, pp. 449–456.

Daumé III, H. & D. Marcu (2004). Generic sentence fusion is an ill-defined summarization task. In *Proceedings of the Text Summarization Branches Out Workshop at ACL-04*, Madrid, Spain, 25–26 July 2004, pp. 96–103.

de Marneffe, M.-C., B. MacCartney & C. D. Manning (2006). Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, Genoa, Italy, 22–28 May 2006, pp. 449–454.

Denis, P. & J. Baldridge (2007). Joint determination of anaphoricity and coreference resolution using integer programming. In *Proceedings of Human Language Technologies Conference of the North American Chapter of the Association for Computational Linguistics,* Rochester, N.Y., 22–27 April 2007, pp. 236–243.

Dorr, B., D. Zajic & R. Schwartz (2003). Hedge trimmer: A parse-and-trim approach to headline generation. In *Proceedings of the Text Summarization Workshop at HLT-NAACL-03,* Edmonton, Alberta, Canada, 2003, pp. 1–8.

Drach, E. (1937). *Grundlagen der deutschen Satzlehre*. Frankfurt/Main, Germany: Diesterweg.

Dras, M. (1997). Reluctant paraphrase: Textual restructuring under an optimisation model. In *Proceedings of the 5th Meeting of the Pacific Association for Computational Linguistics,* Ohme, Japan, 2–5 September, 1997, pp. 98–104.

Edmundson, H. (1969). New methods in automatic extracting. *Journal of the Association for Computing Machinery*, 16(2):264–285. Reprinted in *Advances in Automatic Text Summarization*, Mani, I. and Maybury, M.T. (Eds.), Cambridge, Mass.: MIT Press, 1999, pp.21-42.

Fellbaum, C. (Ed.) (1998). *WordNet: An Electronic Lexical Database*. Cambridge, Mass.: MIT Press.

Filippova, K. & M. Strube (2007a). Generating constituent order in German clauses. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics,* Prague, Czech Republic, 23–30 June 2007, pp. 320–327.

Filippova, K. & M. Strube (2007b). The German Vorfeld and local coherence. *Journal of Logic, Language and Information*, 16(4):465–485.

Filippova, K. & M. Strube (2008a). Dependency tree based sentence compression. In *Proceedings of the 5th International Conference on Natural Language Generation,* Salt Fork, Ohio, 12–14 June 2008, pp. 25–32.

Filippova, K. & M. Strube (2008b). Sentence fusion via dependency graph compression. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing,* Honolulu, Hawaii, 25–27 October 2008, pp. 177–185.

Filippova, K. & M. Strube (2009). Tree linearization in English: Improving language model based approaches. In *Companion Volume to the Proceedings of Human Language Technologies Conference of the North American Chapter of the Association for Computational Linguistics,* Boulder, Colorado, 30 May – 5 June 2009, pp. 225–228.

Firbas, J. (1974). Some aspects of the Czechoslovak approach to problems of functional sentence prespective. In F. Daneš (Ed.), *Papers on Functional Sentence Perspective*, pp. 11–37. Prague: Academia.

Foth, K. & W. Menzel (2006). Hybrid parsing: Using probabilistic models as predictors for a symbolic parser. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, Sydney, Australia, 17–21 July 2006, pp. 321–327.

Frey, W. (2004). A medial topic position for German. *Linguistische Berichte*, 198:153–190.

Gabrilovich, E. & S. Markovitch (2007). Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, Hyderabad, India, 6–12 January 2007, pp. 1606–1611.

Gagnon, M. & L. Da Sylva (2005). Text summarization by sentence extraction and syntactic pruning. In *Proceedings of Computational Linguistics in the North East*, Gatineau, Québec, Canada, 26 August 2005.

Galley, M. & K. R. McKeown (2007). Lexicalized Markov grammars for sentence compression. In *Proceedings of Human Language Technologies Conference of the North American Chapter of the Association for Computational Linguistics*, Rochester, N.Y., 22–27 April 2007, pp. 180–187.

Gazdar, G., E. Klein, G. Pullum & I. Sag (1985). *Generalized Phrase Structure Grammar*. Cambridge, Mass.: Harvard University Press.

Gernsbacher, M. A. & D. J. Hargreaves (1988). Accessing sentence participants: The advantage of first mention. *Journal of Memory and Language*, 27:699–717.

Givón, T. (1983). Topic continuity in spoken English. In T. Givon (Ed.), *Topic Continuity in Discourse: A Quantitative Cross-Language Study*. Amsterdam, The Netherlands: John Benjamins.

Givón, T. (2001). *Syntax: An Introduction*, Vol. I. John Benjamins.

Goodman, J. T. (2001). A bit of progress in language modeling. *Computer Speech and Language*, pp. 403–434.

Graesser, A. C., M. Singer & T. Trabasso (1994). Constructing inferences during narrative text comprehension. *Psychological Review*, 101(3):371–395.

Grefenstette, G. (1998). Producing intelligent telegraphic text reduction to provide an audio scanning service for the blind. In *Working Notes of the Workshop on Intelligent Text Summarization*, Palo Alto, Cal., 23 March 1998, pp. 111–117.

Grosz, B. J., A. K. Joshi & S. Weinstein (1983). Providing a unified account of definite noun phrases in discourse. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, Cambridge, Mass., 15–17 June 1983, pp. 44–50.

Grosz, B. J., A. K. Joshi & S. Weinstein (1995). Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

Gundel, J. K. (1998). Centering theory and the givenness hierarchy: Towards a synthesis. In M. Walker, A. Joshi & E. Prince (Eds.), *Centering in Discourse*, pp. 183–198. Oxford, U.K.: Oxford University Press.

Gundel, J. K., N. Hedberg & R. Zacharski (1993). Cognitive status and the form of referring expressions in discourse. *Language*, 69:274–307.

Hajičová, E., H. Skoumalová & P. Sgall (1995). An automatic procedure for topic-focus identification. *Computational Linguistics*, 21(1):81–94.

Halliday, M. A. K. (1985). *Introduction to Functional Grammar*. London, U.K.: Arnold.

Halliday, M. A. K. & R. Hasan (1976). *Cohesion in English*. London, U.K.: Longman.

Harbusch, K., G. Kempen, C. van Breugel & U. Koch (2006). A generation-oriented workbench for performance grammar: Capturing linear order variability in German and Dutch. In *Proceedings of the 4th International Conference on Natural Language Generation*, Sydney, Australia, 15–16 July 2006, pp. 9–11.

Hatzivassiloglou, V., J. L. Klavans & E. Eskin (1999). Detecting text similarity over short passages: Exploring linguistic feature combinations via machine learning. In *Proceedings of the 1999 SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, College Park, Md., 21–22 June 1999, pp. 203–212.

Hatzivassiloglou, V., J. L. Klavans, M. L. Holcombe, R. Barzilay, M.-Y. Kan & K. R. McKeown (2001). SimFinder: A flexible clustering tool for summarization. In *Proceedings of the 2nd Conference of the North American Chapter of the Association for Computational Linguistics*, Pittsburgh, Penn., 2–7 June 2001, pp. 41–49.

Hawkins, J. A. (1992). Syntactic weight versus information structure in word order variation. In J. Jacobs (Ed.), *Informationsstruktur und Grammatik*, pp. 196–219. Opladen, Germany: Westdeutscher Verlag.

Hockett, C. F. (1958). *A Course in Modern Linguistics*. New York, N.Y.: Macmillan.

Hoffman, B. (1998). Word order, information structure, and centering in Turkish. In M. Walker, A. Joshi & E. Prince (Eds.), *Centering Theory in Discourse*, pp. 251–271. Oxford, U.K.: Oxford University Press.

Hori, C. & S. Furui (2004). Speech summarization: An approach through word extraction and a method for evaluation. *IEEE Transactions on Information and Systems*, E87-D(1):15–25.

Hori, C., S. Furui, R. Malkin, H. Yu & A. Waibel (2003). A statistical approach to automatic speech summarization. *EURASIP Journal on Applied Signal Processing*, 2:128–139.

Hovy, E. H. (2003). Text summarization. In R. Mitkov (Ed.), *The Oxford Handbook of Computational Linguistics*, pp. 583–598. Oxford, U.K.: Oxford University Press.

Jacobs, J. (2001). The dimensions of topic-comment. *Linguistics*, 39(4):641–681.

Jin, R. & A. G. Hauptmann (2003). Automatic title generation for spoken broadcast news. In *Proceedings of the 1st International Conference on Human Language Technology Research*, San Diego, Cal., 18–21 March 2001, pp. 1–3.

Jing, H. (2000). Sentence reduction for automatic text summarization. In *Proceedings of the 6th Conference on Applied Natural Language Processing*, Seattle, Wash., 29 April – 4 May 2000, pp. 310–315.

Jing, H. (2001). *Cut-and-Paste Text Summarization*, (Ph.D. thesis). Computer Science Department, Columbia University, New York, N.Y.

Jurafsky, D. & J. H. Martin (2008). *Speech and Language Processing* (2nd. ed.). Upper Saddle River, N.J.: Prentice Hall.

Karamanis, N., M. Poesio, C. Mellish & J. Oberlander (2004). Evaluating centering-based metrics of coherence for text structuring using a reliably annotated corpus. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, Barcelona, Spain, 21–26 July 2004, pp. 392–393.

Kassner, L., V. Nastase & M. Strube (2008). Acquiring a taxonomy from the German Wikipedia. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*, Marrakech, Morocco, 26 May – 1 June 2008.

Keller, F. (2000). *Gradience in Grammar: Experimental and Computational Aspects of Degrees of Grammaticality*, (Ph.D. thesis). University of Edinburgh.

Kempen, G. & K. Harbusch (2004a). A corpus study into word order variation in German subordinate clauses: Animacy affects linearization independently of grammatical function assignment. In T. Pechmann & C. Habel (Eds.), *Multidisciplinary Approaches to Language Production*, pp. 173–181. Berlin, Germany: Mouton De Gruyter.

Kempen, G. & K. Harbusch (2004b). Generating natural word orders in a semi-free word order language: Treebank-based linearization preferences for German. In A. Gelbukh (Ed.), *Fifth International Conference on Intelligent Text Processing and Computational Linguistics*, pp. 350–354. Berlin, Germany: Springer Verlag.

Kempen, G. & K. Harbusch (2004c). How flexible is constituent order in the midfield of German subordinate clauses? A corpus study revealing unexpected rigidity. In *Proceedings of the International Conference on Linguistic Evidence*, Tübingen, Germany, 29–31 January 2004, pp. 81–85.

Kendall, M. G. (1938). A new measure of rank correlation. *Biometrika*, 30:81–93.

Kintsch, W. & T. van Dijk (1978). Toward a model of text comprehension and production. *Psychological Review*, 85(5):363–394.

Klein, D. & C. D. Manning (2003). Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, Sapporo, Japan, 7–12 July 2003, pp. 423–430.

Klenner, M. (2007). Enforcing consistency on coreference sets. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, Borovets, Bulgaria, 27–29 September 2007, pp. 323–328.

Knight, K. & D. Marcu (2000). Statistics-based summarization – step one: Sentence compression. In *Proceedings of the 17th National Conference on Artificial Intelligence*, Austin, Tex., 30 July – 3 August 2000, pp. 703–711.

Knight, K. & D. Marcu (2002). Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107.

Kohavi, R. & M. Sahami (1996). Error-based and entropy-based discretization of continuous features. In *Proceedings of the 2nd International Conference on Data Mining and Knowledge Discovery*, Portland, Oreg., 2–4 August, 1996, pp. 114–119.

Krahmer, E., E. Marsi & P. van Pelt (2008). Query-based sentence fusion is better defined and leads to more preferred results than generic sentence fusion. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics and Human Language Technologies*, Columbus, Ohio, 15–20 June 2008, pp. 193–196.

Kruijff, G.-J., I. Kruijff-Korbayová, J. Bateman & E. Teich (2001). Linear order as higher-level decision: Information structure in strategic and tactical generation. In *Proceedings of the 8th European Workshop on Natural Language Generation*, Toulouse, France, 6-7 July 2001, pp. 74–83.

Kruijff-Korbayová, I., G.-J. Kruijff & J. Bateman (2002). Generation of appropriate word order. In K. van Deemter & R. Kibble (Eds.), *Information Sharing: Reference and Presupposition in Language Generation and Interpretation*, pp. 193–222. Stanford, Cal.: CSLI.

Kübler, S. & J. Prokic (2006). Why is German dependency parsing more reliable than constituent parsing? In *Proceedings of the 5th Workshop on Treebanks and Linguistic Theories*, Prague, Czech Republic, 1–2 December 2006, pp. 7–18.

Kurz, D. (2000). A statistical account on word order variation in German. In A. Abeillé, T. Brants & H. Uszkoreit (Eds.), *Proceedings of the COLING Workshop on Linguistically Interpreted Corpora*, Luxembourg, 6 August 2000.

Lambrecht, K. (1994). *Information Structure and Sentence Form*. Cambridge, U.K.: Cambridge University Press.

Lamers, M. & H. de Hoop (2005). Animacy information in human sentence processing: An incremental optimization of interpretation approach. In H. Christiansen, P. R. Skadhauge & J. Villadsen (Eds.), *Constraint Solving and Language Processing*, pp. 158–171. Berlin, Germany: Springer.

Langkilde, I. & K. Knight (1998). Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics*, Montréal, Québec, Canada, 10–14 August 1998, pp. 704–710.

Lapata, M. (2003). Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, Sapporo, Japan, 7–12 July 2003, pp. 545–552.

Lapata, M. (2006). Automatic evaluation of information ordering: Kendall's tau. *Computational Linguistics*, 32(4):471–484.

Lemnitzer, L. & C. Kunze (2002). GermaNet – representation, visualization, application. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation*, Las Palmas, Canary Islands, Spain, 29–31 May 2002, pp. 1485–1491.

Levinson, S. C. (1983). *Pragmatics*. Cambridge, U.K.: Cambridge University Press.

Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. In *Proceedings of the Text Summarization Branches Out Workshop at ACL-04*, Barcelona, Spain, 25–26 July 2004, pp. 74–81.

Luhn, H. (1958). The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2:159–165.

Mani, I. (2001). *Automatic Summarization*. Amsterdam, Philadelphia: John Benjamins.

Mani, I. & M. T. Maybury (Eds.) (1999a). *Advances in Automatic Text Summarization*. Cambridge, Mass.: MIT Press.

Mani, I. & M. T. Maybury (1999b). Introduction. In I. Mani & M. T. Maybury (Eds.), *Advances in Automatic Text Summarization*, pp. ix–xv. Cambridge, Mass.: MIT Press.

Mann, W. C. & S. A. Thompson (1988). Rhetorical structure theory. Toward a functional theory of text organization. *Text*, 8(3):243–281.

Manning, C. D. & H. Schütze (1999). *Foundations of Statistical Natural Language Processing*. Cambridge, Mass.: MIT Press.

Marciniak, T. & M. Strube (2005). Discrete optimization as an alternative to sequential processing in NLG. In *Proceedings of the 10th European Workshop on Natural Language Generation*, Aberdeen, U.K., 8–10 August, 2005, pp. 101–108.

Marsi, E. & E. Krahmer (2005). Explorations in sentence fusion. In *Proceedings of the 10th European Workshop on Natural Language Generation*, Aberdeen, U.K., 8–10 August 2005, pp. 109–117.

Marsi, E. & E. Krahmer (2007). Annotating a parallel monolingual treebank with semantic similarity relations. In *Proceedings of the 6th Workshop on Treebanks and Linguistic Theories*, Bergen, Norway, 7–8 December 2007.

McDonald, R. (2006). Discriminative sentence compression with soft syntactic evidence. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy, 3–7 April 2006, pp. 297–304.

McNamara, D. S., E. Kintsch, N. B. Songer & W. Kintsch (1996). Are good texts always better? Interactions of text coherence, background knowledge, and levels of understanding in learning from text. *Cognition and Instruction*, 14(1):1–43.

Mel'čuk, I. A. (2003). Levels of dependency in linguistic description: Concepts and problems. In V. Agel, L. Eichinger, H.-W. Eroms, P. Hellwig, H. Heringer & H. Lobin (Eds.),

Dependency and Valency. An International Handbook of Contemprorary Research, Vol. 1, pp. 188–229. Berlin - New York: de Gruyter.

Meyers, A., R. Yangarber & R. Grishman (1996). Alignment of shared forests for bilingual corpora. In *Proceedings of the 16th International Conference on Computational Linguistics*, Copenhagen, Denmark, 5–9 August 1996, pp. 460–465.

Milne, D. & I. H. Witten (2008). An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In *Proceedings of the AAAI 2008 Workshop on Wikipedia and Artificial Intelligence*, Chicago, IL, 13-14 July, 2008.

Molnár, V. (1991). *Das TOPIK im Deutschen und im Ungarischen*. Stockholm, Sweden: Almqvist and Wiksell.

Morris, J. & G. Hirst (1991). Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics*, 17(1):21–48.

Müller, C. & M. Strube (2006). Multi-level annotation of linguistic data with MMAX2. In S. Braun, K. Kohn & J. Mukherjee (Eds.), *Corpus Technology and Language Pedagogy: New Resources, New Tools, New Methods*. Peter Lang: Frankfurt a.M., Germany.

Müller, G. (1999). Optimality, markedness, and word order in German. *Linguistics*, 37:777–818.

Nastase, V. & M. Strube (2008). Decoding Wikipedia category names for knowledge acquisition. In *Proceedings of the 23rd Conference on the Advancement of Artificial Intelligence*, Chicago, Ill., 13–17 July 2008, pp. 1219–1224.

Nelken, R. & S. M. Shieber (2006). Towards robust context-sensitive sentence alignment for monolingual corpora. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy, 3–7 April 2006, pp. 161–168.

Nguyen, M. L., A. Shimazu, S. Horiguchi, B. T. Ho & M. Fukushi (2004). Probabilistic sentence reduction using support vector machines. In *Proceedings of the 20th International Conference on Computational Linguistics*, Geneva, Switzerland, 23–27 August 2004, pp. 743–749.

Ouhalla, J. (1994). *Introducing Transformational Grammar: From Rules to Principles and Parameters*. Edward Arnold.

Pappert, S., J. Schließ er, D. P. Janssen & T. Pechmann (2007). Corpus- and psycholinguistic investigations of linguistic constraints on German word order. In A. Späth (Ed.), *Interfaces and interface conditions*, pp. 299–328. de Gruyter.

Poesio, M., R. Stevenson, B. Di Eugenio & J. Hitzeman (2004). Centering: A parametric theory and its instantiations. *Computational Linguistics*, 30(3):309–363.

Ponzetto, S. P. & R. Navigli (2009). Large-scale taxonomy mapping for restructuring and integrating Wikipedia. In *Proceedings of the 21th International Joint Conference on Artificial Intelligence*, Pasadena, Cal., 14–17 July 2009. To appear.

Ponzetto, S. P. & M. Strube (2007a). Deriving a large scale taxonomy from Wikipedia. In *Proceedings of the 22nd Conference on the Advancement of Artificial Intelligence*, Vancouver, B.C., Canada, 22–26 July 2007, pp. 1440–1445.

Ponzetto, S. P. & M. Strube (2007b). Knowledge derived from Wikipedia for computing semantic relatedness. *Journal of Artificial Intelligence Research*, 30:181–212.

Postolache, O., I. Kruijff-Korbayová & G.-J. Kruijff (2005). Data-driven approaches for information structure identification. In *Proceedings of the Human Language Technology Conference and the 2005 Conference on Empirical Methods in Natural Language Processing*, Vancouver, B.C., Canada, 6–8 October 2005, pp. 9–16.

Prince, A. & P. Smolensky (2004). *Optimality Theory: Constraint Interaction in Generative Grammar*. Blackwell Publishers.

Prince, E. F. (1981). Towards a taxonomy of given-new information. In P. Cole (Ed.), *Radical Pragmatics*, pp. 223–255. New York, N.Y.: Academic Press.

Prince, E. F. (1999). How not to mark topics: 'Topicalization' in English and Yiddish. *Texas Linguistics Forum*.

Punyakanok, V., D. Roth, W.-t. Yih & D. Zimak (2004). Semantic role labeling via integer linear programming inference. In *Proceedings of the 20th International Conference on Computational Linguistics*, Geneva, Switzerland, 23–27 August 2004, pp. 1346–1352.

Radev, D. R., E. H. Hovy & K. R. McKeown (2002). Introduction to the special issue on summarization. *Computational Linguistics*, 28(4):399–408.

Radev, D. R. & K. R. McKeown (1998). Generating natural language summaries from multiple on-line sources. *Computational Linguistics*, 24(3):469–500.

Rambow, O. (1993). Pragmatic aspects of scrambling and topicalization in German. In *Workshop on Centering Theory in Naturally-Occurring Discourse*. Institute for Research in Cognitive Science (IRCS), Philadelphia, Penn.: University of Pennsylvania, May 1993.

Ratnaparkhi, A. (2000). Trainable methods for surface natural language generation. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics*, Seattle, Wash., 29 April – 3 May 2000, pp. 194–201.

Reinhart, T. (1981). Pragmatics and linguistics. An analysis of sentence topics. *Philosophica*, 27(1):53–94.

Reiter, E. & R. Dale (2000). *Building Natural Language Generation Systems*. Cambridge, U.K.: Cambridge University Press.

Riedel, S. & J. Clarke (2006). Incremental integer linear programming for non-projective dependency parsing. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, Sydney, Australia, 22–23 July 2006, pp. 129–137.

Riezler, S., T. H. King, R. Crouch & A. Zaenen (2003). Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for Lexical-Functional Grammar. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, Edmonton, Alberta, Canada, 27 May –1 June 2003, pp. 118–125.

Ringger, E., M. Gamon, R. C. Moore, D. Rojas, M. Smets & S. Corston-Oliver (2004). Linguistically informed statistical models of constituent structure for ordering in sentence realization. In *Proceedings of the 20th International Conference on Computational Linguistics*, Geneva, Switzerland, 23–27 August 2004, pp. 673–679.

Ross, J. R. (1967). *Constraints on variables in syntax*, (Ph.D. thesis). MIT.

Roth, D. & W.-t. Yih (2004). A linear programming formulation for global inference in natural language tasks. In *Proceedings of the 8th Conference on Computational Natural Language Learning*, Boston, Mass., USA, 6–7 May 2004, pp. 1–8.

Schmid, H. (1997). Probabilistic Part-of-Speech tagging using decision trees. In D. Jones & H. Somers (Eds.), *New Methods in Language Processing*, pp. 154–164. London, U.K.: UCL Press.

Sgall, P., E. Hajičová & J. Panevová (1986). *The Meaning of the Sentence in Its Semantic and Pragmatic Aspects*. Dordrecht, The Netherlands: D. Reidel.

Skorochod'ko, E. (1972). Adaptive method of automatic abstracting and indexing. In *In* Information Processing 71: Proceedings of the IFIP Congress 71, pp. 1179–1182.

Sleator, D. & D. Temperley (1993). Parsing English with a link grammar. In *3rd International Workshop on Parsing Technologies*.

Spärck Jones, K. (1999). Automatic summarizing: Factors and directions. In I. Mani & M. T. Maybury (Eds.), *Advances in Automatic Text Summarization*, pp. 1–12. Cambridge, Mass.: MIT Press.

Spärck-Jones, K. (2007). *Automatic summarising: A review and discussion of the state of the art*. Technical Report UCAM-CL-TR-679, Cambridge, U.K.: University of Cambridge, Computer Laboratory.

Späth, H. (1985). *Cluster Dissection and Analysis: Theory, FORTRAN Programs, Examples*. Chichester, U.K.: Ellis Horwood.

Speyer, A. (2005). Competing constraints on Vorfeldbesetzung in German. In *Proceedings of the Constraints in Discourse Workshop*, Dortmund, 3–5 July 2005, pp. 79–87.

Strawson, P. F. (1964). Identifying reference and truth-values. In D. Steinberg & L. Jacobovits (Eds.), *Semantics*, pp. 86–99. Cambridge, U.K.: Cambridge University Press.

Strube, M. & S. P. Ponzetto (2006). WikiRelate! Computing semantic relatedness using Wikipedia. In *Proceedings of the 21st National Conference on Artificial Intelligence*, Boston, Mass., 16–20 July 2006, pp. 1419–1424.

Telljohann, H., E. W. Hinrichs & S. Kübler (2003). *Stylebook for the Tübingen treebank of written German (TüBa-D/Z)*. Technical Report: Seminar für Sprachwissenschaft, Universität Tübingen, Tübingen, Germany.

Turner, J. & E. Charniak (2005). Supervised and unsupervised learning for sentence compression. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, Ann Arbor, Mich., 25–30 June 2005, pp. 290–297.

Uchimoto, K., M. Murata, Q. Ma, S. Sekine & H. Isahara (2000). Word order acquisition from corpora. In *Proceedings of the 18th International Conference on Computational Linguistics*, Saarbrücken, Germany, 31 July – 4 August 2000, pp. 871–877.

Uszkoreit, H. (1987). *Word Order and Constituent Structure in German*. CSLI Lecture Notes. Stanford: CSLI.

Vallduví, E. & E. Engdahl (1996). The linguistic realization of information packaging. *Linguistics*, 34:459–519.

Velldal, E. & S. Oepen (2006). Statistical ranking in tactical generation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, Sydney, Australia, 22–23 July 2006, pp. 517–525.

Versley, Y. (2005). Parser evaluation across text types. In *Proceedings of the 4th Workshop on Treebanks and Linguistic Theories*, Barcelona, Spain, 9-10 December 2005.

Wan, S., R. Dale, M. Dras & C. Paris (2005). Searching for grammaticality and consistency: Propagating dependencies in the Viterbi algorithm. In *Proceedings of the 10th European Workshop on Natural Language Generation*, Aberdeen, U.K., 8–10 August 2005, pp. 211–216.

Wan, S., R. Dale, M. Dras & C. Paris (2008). Seed and grow: Augmenting statistically generated summary sentences using schematic word patterns. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, Honolulu, Hawaii, 25–27 October 2008, pp. 543–552.

Wan, S., M. Dras, R. Dale & C. Paris (2009). Improving grammaticality in statistical sentence generation: Introducing a dependency spanning tree algorithm with an argument satisfaction model. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, Athens, Greece, 30 March – 3 April 2009 , pp. 852–860.

Webber, B. L., M. Stone, A. Joshi & A. Knott (2003). Anaphora and discourse structure. *Computational Linguistics*, 29(4):545–588.

Weber, A. & K. Müller (2004). Word order variation in German main clauses: A corpus analysis. In *Proceedings of the 5th International Workshop on Linguistically Interpreted Corpora*, 29 August, 2004, Geneva, Switzerland, pp. 71–77.

Wu, Z. & M. Palmer (1994). Verb semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, Las Cruces, N.M., 27–30 June 1994, pp. 133–138.

KATJA FILIPPOVA: CURRICULUM VITAE

Technical University of Darmstadt: PhD, Computational Linguistics
January 2006 – October 2009, Darmstadt, Germany

University of Tübingen: MA, Computational Linguistics
October 2003 – October 2005, Tübingen, Germany

State University of St. Petersburg: BA, Theoretical Linguistics
September 1999 – July 2003, St. Petersburg, Russia

Anichkov Lyceum: High school dimploma
September 1996 – June 1999, St. Petersburg, Russia